ASPERA-3 EGSE User Manual

	Name and function	**Date**	**Signature**
Prepared by:	Riccardo Cerulli-Irelli		
Verified by:	Herman Andersson, EM		
Approved by:	Stas Barabash, Co-PI		

Distribution List:

IRF	Stas Barabash	
	Herman Andersson	
FMI	Tuukka Sales	
IFSI	Stefano Orsini	
	Anna Milillo	
	Alessando Mura	
	Andrea Mattana	

Change Record:

Version	Date	Changed Paragraphs	Remarks
Issue 1, Rev 3	5/3/2001	4, 5	Removed (was empty)
Issue 1, Rev 3	5/3/2001	3.2	Written (was empty)
Issue 1, Rev 4	15/2/2002	2.2	Added powConvrc configuration file

Reference documents

Document Reference	Name	Reference
RD1	External interfaces for the Rosetta CCS	RO-TER-IS-4001 Issue 1.5

Acronyms

ADC	Analog to Digital Converter
APID	Application Process IDentification
CAT	Packet CATegory
CCS	Central Check-out System
CNR	Consiglio Nazionale delle Ricerche
CPU	Control Processing Unit
DPU	Digital Processing Unit
DSP	Digital Signal Processor
EGSE	Electrical Ground Support Equipment
ESA	European Space Agency
ILT	Instrument Level Test
HK	HouseKeeping
HW	HardWare
ID	Identification
I/F	Interface
IFSI	Istituto di Fisica dello Spazio Interplanetario
MOC	Mission Operation Center
NA	Not Applicable
OBDH	On Board Data Handling
PID	Process ID
RDx	Reference Document x [RDx]
S/C	Spacecraft
SDM	Science Display Module
SCOE	System Check-Out Equipment
SLT	System Level Test
TBC	To Be Confirmed
TBD	To Be Defined
TBW	To Be Written
TC	Telecommand
TM	Telemetry

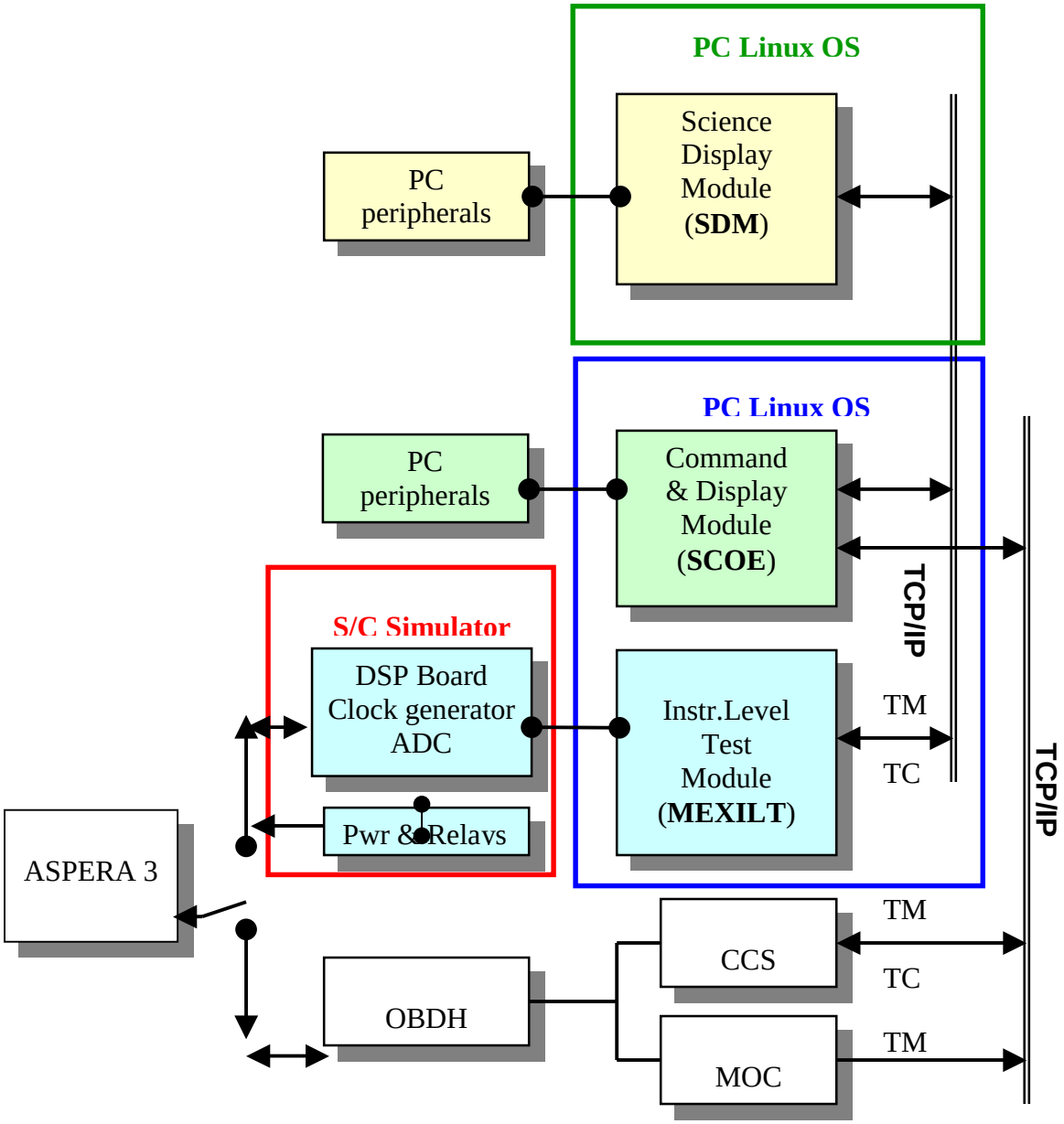
1 General Description 5
1.1 SCOE 6
1.2 MEXILT 6
1.3 S/C simulator 6
1.4 SDM 7
1.5 Instrument Interface 8
1.6 Stimulator Interface 8
1.7 User Interface 8
1.8 Network Interface 9
1.9 Software Description 9
2 Egse Configuration 10
2.1 Directory tree 10
2.2 Run Time Configuration 12
2.3 Instrument/System Level Test 14
2.4 Instrument Level Test EGSE 14
2.4.1 Physical configuration 15
2.4.2 Software 16
2.5 System Level Test EGSE 18
2.5.1 Physical configuration 18
2.5.2 Software 19
3 Operations 21
3.1 Instrument Level Test Operations 21
3.1.1 Program setup 21
3.1.2 SCOE interface 21
3.1.3 Command List 26
3.1.4 Archiving 38
3.1.5 Playback 39
3.2 System Level Test Operations 40
3.2.1 Program Setup 40
3.2.2 SCOE Interface 40
3.2.3 Command List 41
3.2.4 Commands sequence 45
4 APPENDIX 46
4.1 External Electronic Schematics 46
4.2 ASPERA High Level Command ILT 51
4.3 ASPERA High Level Command SLT 52

1 General Description

The ASPERA 3 EGSE consists of two IBM-PC class computers running Linux operating system. The first PC hosts two programs modules linked via standard TCP/IP socket:

- The "Instrument Level Test" module (**MEXILT**) to be used only at instrument level test
- The "Command and Display Module (**SCOE**) to be used both at instrument and system level test.

The second PC, hosting the "science Display Module" (**SDM**), is connected via NFS to the SCOE, and is responsible for science display of the telemetry data archived by the SCOE.
The EGSE block diagram is shown in the following figure.



1.1 SCOE

The SCOE module is responsible for command sequence generation and housekeeping data display. This module, during the instrument level tests, is connected to MEXILT using the same (Internet domain stream socket) protocol used by CCS in order to be used with only minor modifications at system level test. At system level test this module is connected to CCS.

1.2 MEXILT

The MEXILT module is connected to the ASPERA instrument through a S/C simulator board, which provide the OBDH hardware interface. The MEXILT module provides the basic OBDH-CCS emulation and is responsible for TLM/TLC serialization, verification and transmission, and is command controlled by the SCOE module linked via TCP/IP connection. Due to this inter process communication scheme, the two program modules may run on different computers connected to Internet.

1.3 S/C simulator

The S/C simulator is designed around the TI 320C54x DSKplus board plus a simple external electronic board for signal generation and OBDH compliant interfaces.

The main characteristics of the TI 320C54x board (shown in fig 5.1.2-2) are:

- One TMS320C542 ('C542) 40 MHz enhanced fixed-point DSP
- 40 MIPS (25-ns instruction cycle time)
- 10K words of dual-access RAM (DARAM)
- 2K words boot ROM
- One time-division-multiplexed (TDM) serial port
- One buffered serial port (BSP)
- One host port interface (HPI) for PC-to-DSP communications
- One on-chip timer
- Programmable, voice-quality TLC320AC01 (DAC, ADC interface circuit)
- I/O expansion bus and control signals for external designs

Figure 1–1. DSKplus Board Diagram

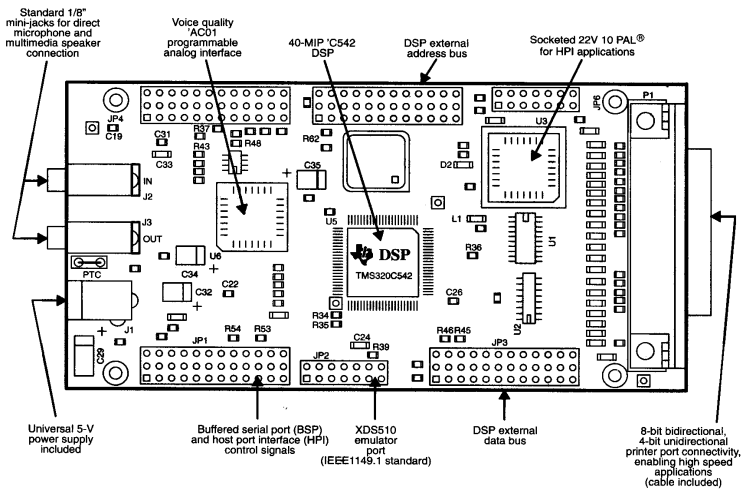

Standard 1/8" mini-jacks for direct microphone and multimedia speaker connection
Voice quality 'AC01 programmable analog interface
40-MIP 'C542 DSP
DSP external address bus
Socketed 22V 10 PAL® for HPI applications
DSP
TMS320C542
Universal 5-V power supply included
Buffered serial port (BSP) and host port interface (HPI) control signals
XDS510 emulator port (IEEE1149.1 standard)
DSP external data bus
8-bit bidirectional, 4-bit unidirectional printer port connectivity, enabling high speed applications (cable included)

Figure 1 – The TI 320C54x DSKplus board

This board connect to the (MEXILT) module through the PC parallel port at a speed of ~ 100 Kbytes/s.

At start up the MEXILT module download the (C language) program to the DSK. The DSK receive the TC packets and local instruction and send TM packets from/to ILT PC.
The figure 1 shows the overall S/C simulator concept, "external electronic" schematics are in Appendix.

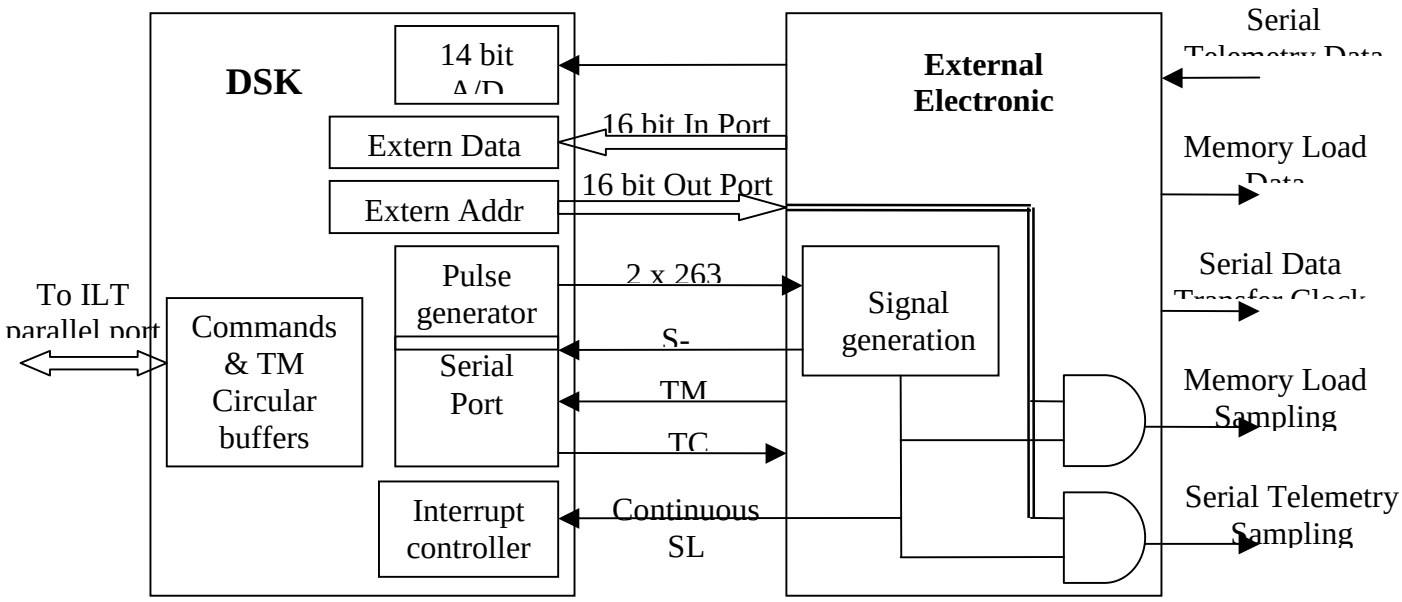


Figure 5.1.2-3 – **The S/C simulator concept**

1.4 SDM

1.5 Instrument Interface

The EGSE will be connected only to the ASPERA-3 Main Unit of the ASPERA-3 experiment, and this connection is realised through the spacecraft simulator interface. This interface surface will be active in all operation foreseen for the EGSE. The spacecraft interface will be simulated by the EGSE, and be fully compliant with the spacecraft bus system (including the connectors). The spacecraft power simulator, included in the S/C simulator unit, will provide power and switching functions to the ASPERA-3 experiment. The power supply will overvoltage and current limiting capabilities. The power will be interfaced to the ASPERA-3 through one connector.

The EGSE consists of an RTU simulator for controlling the RPC and for acquiring the housekeeping and science TM information. The science data may contain also memory dumps, or other engineering data. The EGSE will acquire through the spacecraft communication interface the telemetry packets, the EGSE will simulate the rate at which the telemetry packets are acquired from ASPERA-3 instrument, including multiple telemetry rates.

All command to the instrument will be via the spacecraft serial interface. The interface simulator is a part of the EGSE. The EGSE will simulate the distribution of synchronisation and timing information.

For the ASPERA-3 experiment there will be two types of data that will be acquired by the EGSE: housekeeping and science. The science data may contain memory dumps, or other engineering data. For controlling the Spacecraft Powered Thermistor, a simulator of the Onboard Monitoring System (OMS) is implemented in the S/C simulator (ADC etc.).

1.6 Stimulator Interface

The ASPERA-3 instrument will not use any external or internal stimuli and due to this there is no stimuli interface of the EGSE.

1.7 User Interface

The user interface of the EGSE/S is the keyboard, mouse and the monitor's display. One adequately sized monitor will be available for tabular and graphical data display. Optionally other bus monitors may be required, their operation will be supported, but the additional monitors are not part of the EGSE. The EGSE is capable of displaying decommutated data products from the sensors as the data is being acquired in real time mode of operation (when the EGSE is directly receiving data from the ASPERA-3). The EGSE will be capable to analyse data recorded previously, the amount of historical data available is limited to the amount of temporary data storage available and to the amount of data accessible via the networks. It will be possible to display replayed plots in conjunction with the real time data.

There will be capability to execute single commands and data loads from the EGSE. Hazardous commands shall have an extra level of protection to allow the user extra opportunities to abort sending them. There will be a capability to execute automatic test sequences by the EGSE, such as data selection and limit verification, etc.

The EGSE will record in an ASCII file the history of events that occurred during its operation. Any events that are pertinent to instrument operations shall be logged. Optionally other PCs for monitoring the data exchange (bus monitors) may be required. Their operation will be supported, but they are not part of the EGSE.

1.8 Network Interface

During its operation the EGSE will be connected both to LAN's and WAN's. It is foreseen that the EGSE will analyse data obtained via a network during the flight operation phase, or that the data storage will be organised via network. The EGSE will support these functions. During system level tests the EGSE will be connected to the central checkout system, provided by the project. The Central Checkout System (CCS) is not a part of the EGSE. It is assumed that the communication between the EGSE and the CCS shall use the network interface.

1.9 Software Description

The two SW modules, written in C++, will run under the Linux operating system (Slakware distribution). The SCOE module will use the KDE-Qt X Library and will communicate with MEXILT module using BSD socket communication protocol. The user interface will be a graphical windows interface.
The S/C simulator SW is written in C language and downloaded to the DSK board from MEXILT module at start up.

2 Egse Configuration

2.1 Directory tree

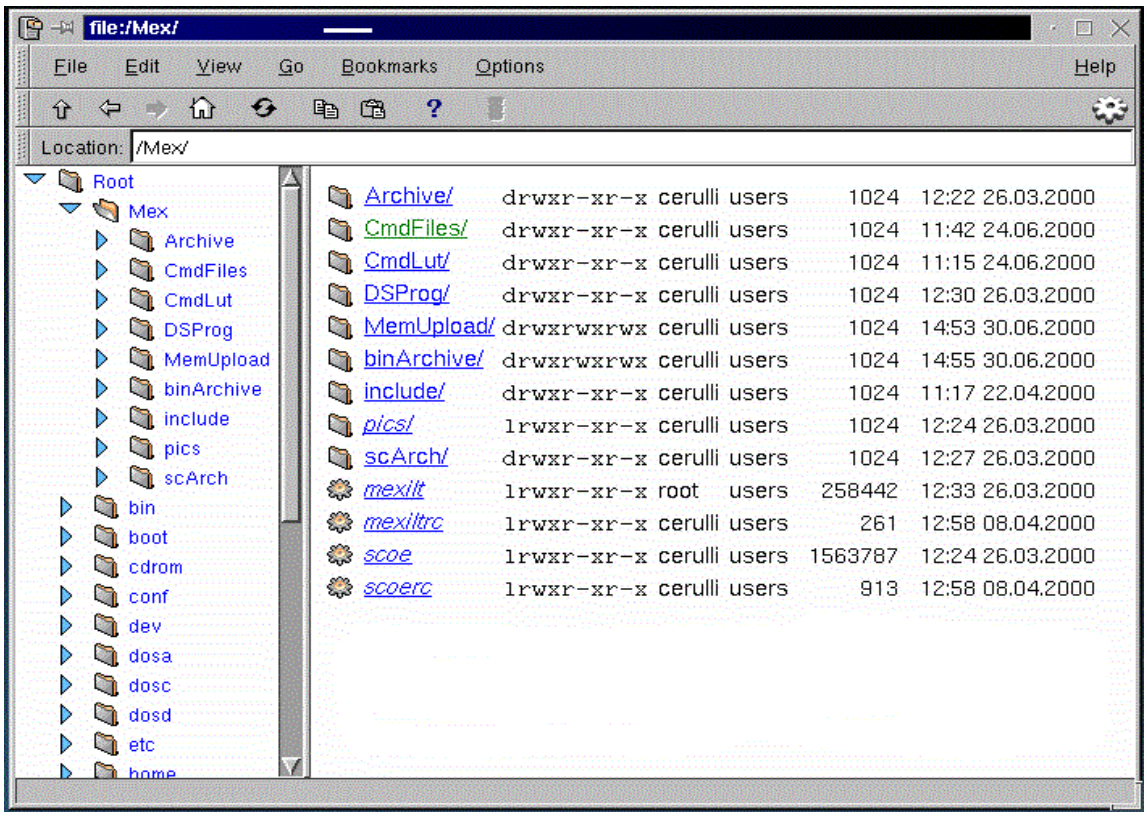
The full EGSE software is contained in /Mex directory. File names in italic shows links to other locations, the real files are maintained by CVS.

Directory list in /Mex:

Archive	Configurable archive directory
CmdFiles	Configurable directory for commands procedure and lookup tables
CmdLut	Configurable directory for commands procedure and lookup tables
DSProg	For internal use
MemUpload	Contains DPU binary files to upload
binArchive	Archive directory for old version of EGSE programs
include	For internal use
scArch	Configurable S/C archive directory

File list in /Mex:

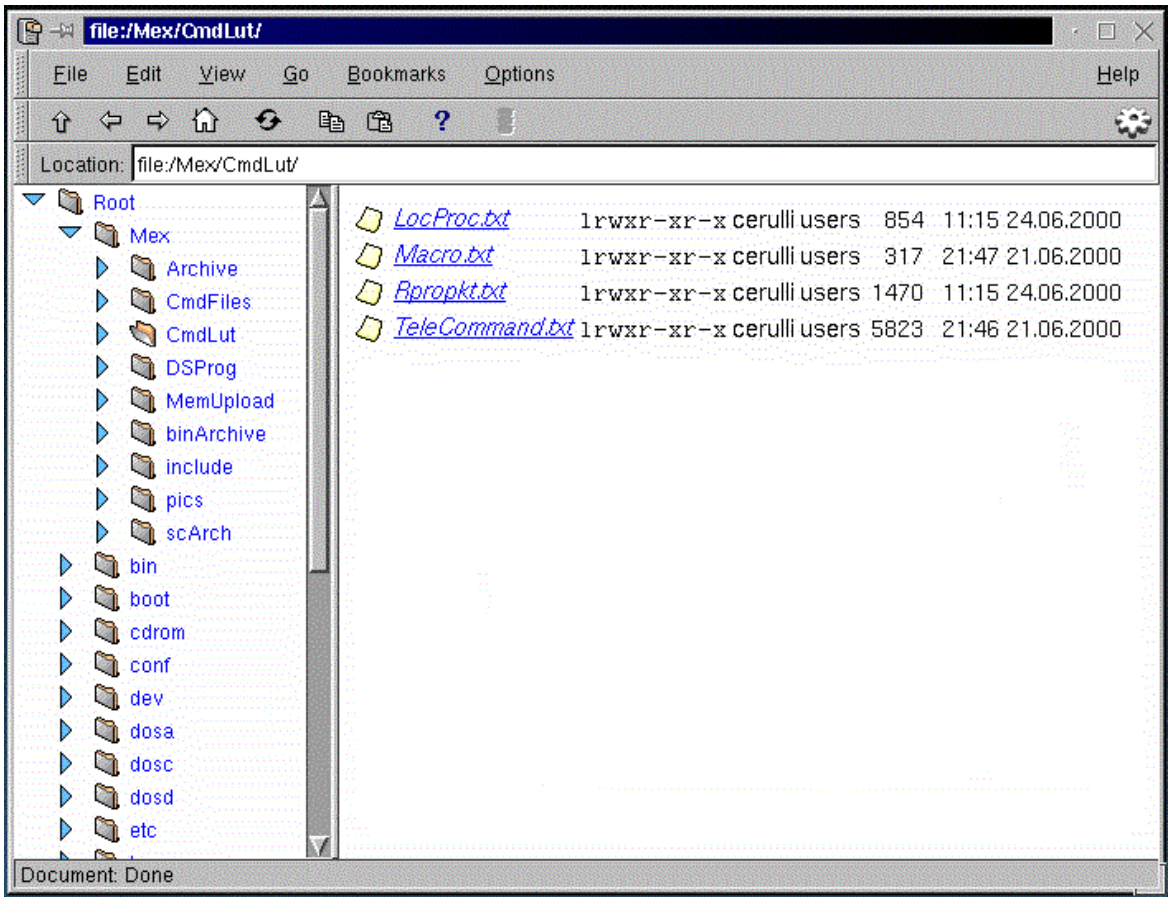
scoe	Main EGSE program for sending telecommands and receiving telemetry.
scoerc	Run time configuration for scoe.
mexilt	Spacecraft simulator program.
mexiltrc	Run time configuration for mexilt.
PowConvrc	Engineering unit conversion constants



This figure shows the CmdFile directory that now contains (links) to some lookup table .

File list in /Mex:/CmdLut

LocProc.txt	Pseudo programming language definitions.
Macro.txt	Macro expansion file. See Macro command chapter.
Rpropkt.txt	CCS protocol commands.
Telecommand.txt	High level command file. See relevant chapter.



2.2 Run Time Configuration

The 3 files **scoerc powConvrc** and **mexiltrc** contains initial configuration read at run time

File location: /Mex/*scoerc* **Link to** /users/cerulli/cpp/scoe/scoe/scoerc

```
# SCOE Configuration file File. Roma 23.5.00

[Directory]
    #-------- TM archive directory
TLMARCH=/Mex/Archive/

    #-------- Command archive directory
CMDARCH =/Mex/Archive/

    #-------- Command directory
CMDFILES =/Mex/CmdFiles/

    #-------- Load memory binary files directory
BINFILES =/Mex/MemUpload/

    #-------- TeleCommand acronym file at ILT
TLCFILE =/Mex/CmdLut/TeleCommand.txt

    #-------- TeleCommand macro file at ILT
MACROFILE=/Mex/CmdLut/Macro.txt

    #-------- TeleCommand acronym file at SLT
TLCFILESLT =/Mex/CmdLut/TeleCommandSLT.txt

    #-------- TeleCommand macro file at SLT
MACROFILESLT=/Mex/CmdLut/MacroSLT.txt

    #-------- RPRO macro file
RPROFILE=/Mex/CmdLut/Rpropkt.txt

    #-------- Local procedure pseudo instruction file
LPROCTBL=/Mex/CmdLut/LocProc.txt

    #-------- Command list include file
CMDTBL=/Mex/include/dspcmtbl.h

[IntConst]
    #-------- S/C simulator model (0 .. 2)
SIMODEL=1

    #-------- Socket port number
PORT=6010

    #-------- Seconds to reopen TC archive
SEC_TO_NEW_ARCTC=3600

    #-------- Seconds to reopen TM archive
SEC_TO_NEW_ARCTM=3600
```

File location: /Mex/*powConvrc* **Link to** /users/cerulli/cpp/scoe/scoe/ powConvrc

```
#####################################
# Power conversion configuration file
#####################################
#
# The conversion is X= AY + B with
# Y=byte/word value, X=value in engineering unit
# In this file are the A and B coefficients for the various
# tabbed panels in the Status/HK windows

# Power panel (astatus7)
[Power]
    #-------- I30
AI30=0.1176
BI30=0.0
    #-------- I5
AI5=0.0196
BI5=0.0
    #-------- VM12
AVM12=-0.047
BVM12=0.0
    #-------- VM5
AVM5=-0.0196
BVM5=0.0
    #-------- V5
AV5=0.0196
BV5=0.0
    #-------- V12
AV12=0.047
BV12=0.0
    #-------- V30
AV30=0.1176
BV30=0.0

    #------- Scanner panel
[ScanPos]
    #------- pos to degree conversion  (astatus6)
ADEG=0.772
BDEG=0.0
```

File location: /Mex/*mexiltrc* **Link to** /users/cerulli/cpp/mexilt/mexilt/mexiltrc

```
# mexilt Configuration file File

[String]
    #-------- TM archive directory
FILARCH=/Mex/scArch/
    #-------- Socket server
SERVERHOST=127.0.0.1

[IntConst]
    #-------- Socket port number
PORT=6010
    #-------- timeout for requested ack
ACKATIMOUT=20
```

2.3 Instrument/System Level Test

The SCOE module is used both at instrument and system level test. The few differences in the two configurations are resolved at compile time commenting/uncommenting the constant variable ILT found in

```
/users/cerulli/cpp/scoe/scoe/configura.h
```

```
#ifndef __320c542_H__
#define __320c542_H__

/* --- Conditional compilation --- */

//#define ILT                  // Using ILT client

#ifndef ILT
    #define NET_BYTE_ORDER     // Use net byte order (big endian)
#endif
.. ..
.. ..
```

The MEXILT and S/C sim modules are used only at ILT, while SDM is used in both configurations.

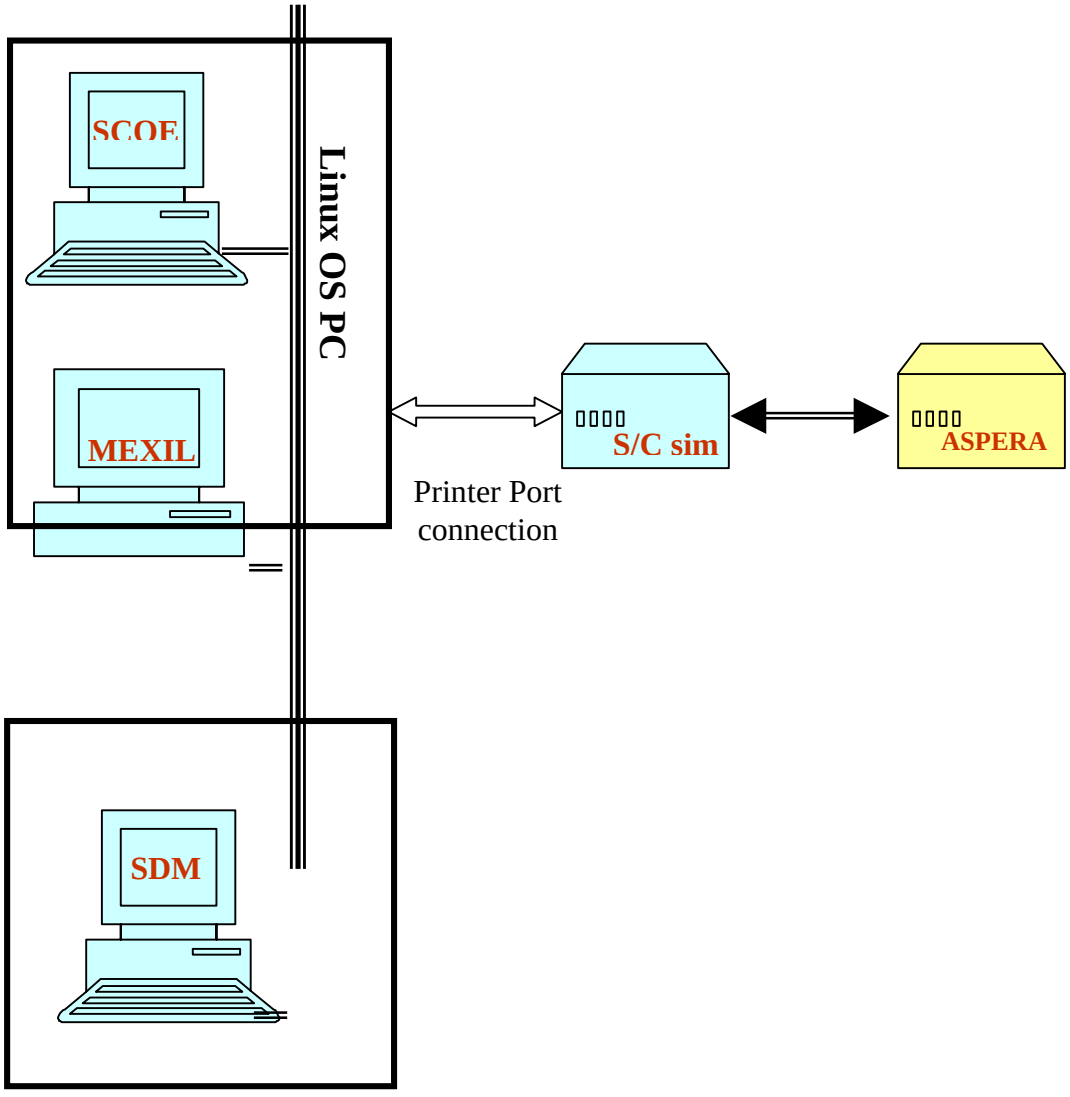
2.4 Instrument Level Test EGSE

In this configuration both the SCOE and MEXILT modules are used. The SCOE program must be compiled with (defining) the ILT constant in

```
/users/cerulli/cpp/scoe/scoe/configura.h

/* --- Conditional compilation --- */

#define ILT                  // Using ILT client
```

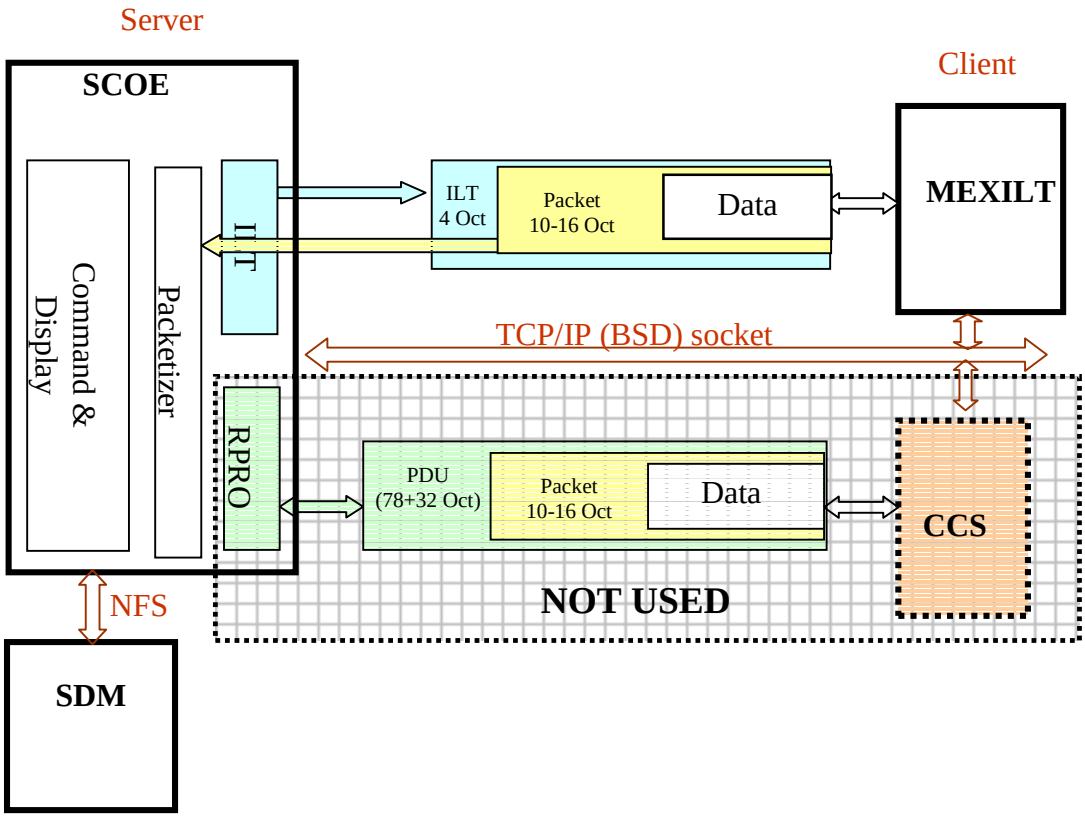
2.4.1 Physical configuration

The SCOE and MEXILT programs runs both on a single PC, and the Science Display Module (SDM) run on a separate PC sharing (via NFS) with SCOE/MEXILT PC the telemetry archive directory structure



2.4.2 Software

SCOE communicates with **MEXILT** through TCP/IP connection, the last communicates with **S/C SIM** through the parallel port. The S/C SIM is connected to ASPERA3 via standard TM/TC interfaces. SCOE/MEXILT PC communicates with the Science Display Module (**SDM**) PC sharing (via NFS) the telemetry archive directory structure.



2.4.2.1 SCOE

The SCOE program is the main EGSE interface. The program consists of an editor-like window used to type, edit and send commands to MEXILT. Commands (one line per command) can be sent one by one, or as a sequence at a defined time interval.
Instructions can't be used inside a command file transmitted directly as with TXFILE command.

ASPERA 3 telemetry packets received by MEXILT can be displayed in tabular mode, and the housekeeping packets are displayed in appropriate windows.
All ASPERA 3 telemetry packets can be stored in the archive directory for the SDM elaboration.

2.4.2.2 MEXILT

MEXILT program simulate the MEX spacecraft and is responsible to deliver to ASPERA the telecommands received by SCOE and to deliver to SCOE the telemetry received by ASPERA.

MEXILT program consists of 3 concurrent threads:

- A thread responsible of TCP/IP communications with SCOE
- A thread responsible of continuous telemetry monitoring
- An execution thread interpreting and executing the SCOE commands

Commands received from SCOE are coded as a "packet" with:

- Word 1 = ID identify the command
- Word 2 = N number of words to follow
- Word 3 = Data parameters (if any)
- . . . . . .
- Word N+2= Last parameter

The commands received from SCOE are divided in 3 types, defining the execution priority from high to low.
The commands are stored in 3 different circular buffers

- **Immediate** commands
- **Program** commands
- **Normal** commands

Generally each command can belong to each of the 3 types.

Immediate commands are executed at the end of the current command execution phase.

Program commands are commands executed as a sequence. The sequence (i.e. the program) is formed by standard commands plus a few "pseudo instructions" special commands, defining elementary programming language statements (i.e. for loop, if statements, setting of program variables etc).

Normal commands are executed when no other command type is present.

2.4.2.3 SDM

2.5 System Level Test EGSE

In this configuration only the SCOE module is used. The SCOE program must be compiled without (commenting) the ILT constant in

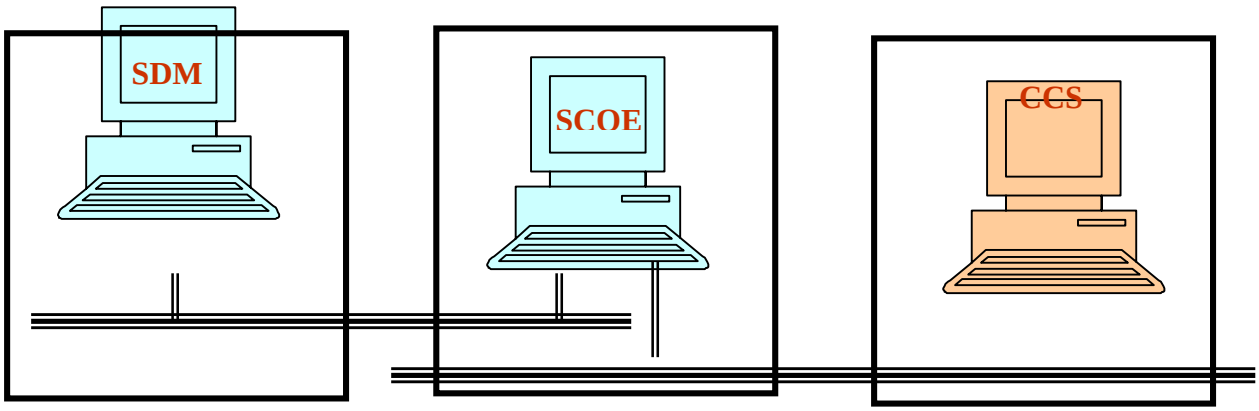
```
/users/cerulli/cpp/scoe/scoe/configura.h

/* --- Conditional compilation --- */

//#define ILT                    // Using ILT client
```

To properly accept the time synchronization request from CCS, the “mode” of the system “date” program must be changed (as super user) to:

```
chmod u+s /bin/date
```

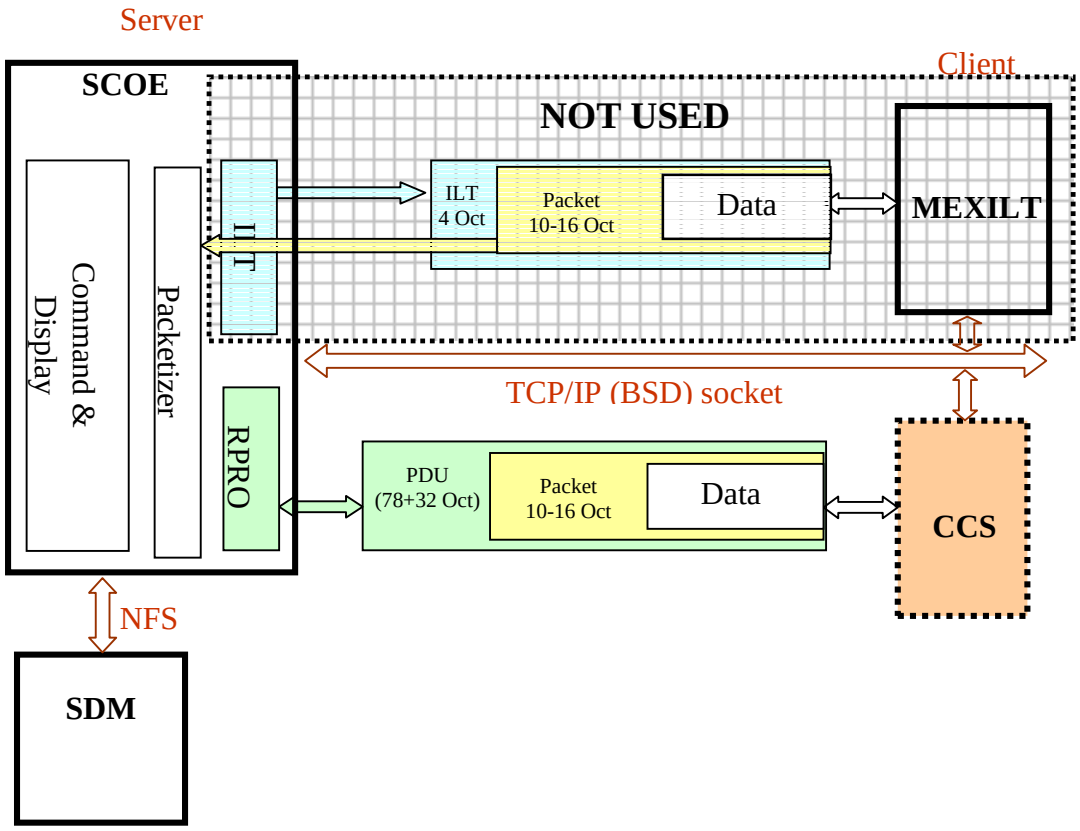
2.5.1 Physical configuration

The SCOE and the Science Display Module (SDM) run on a separate PC sharing (via NFS) with the telemetry archive directory structure



2.5.2 Software

SCOE communicates with the ESA CCS through TCP/IP connection, and with the Science Display Module (SDM) PC sharing, via a second network interface using TCP/IP NFS protocol, the telemetry archive directory structure.



2.5.2.1 SCOE

The SCOE program is the main EGSE interface. The program consists of an editor-like window used to type, edit and send commands to the CCS. Commands (one line per command) can be sent one by one, or as a sequence at a defined time interval.

ASPERA 3 telemetry packets received by CCS can be displayed in tabular mode, and the housekeeping packets are displayed in appropriate windows.
All ASPERA 3 telemetry packets can be stored in the archive directory for the SDM elaboration.

2.5.2.2 SDM

3 Operations

The ASPERA EGSE functionalities are very similar at both Instrument Level Test (ILT) and System Level Test (SLT). The functionalities consist essentially in sending single commands, commands sequences and receiving telemetry.
The telemetry received may be selectively shown in tabular form, interpreted and shown in appropriate windows (housekeeping), interpreted and used in command sequences. All the communications in/out with the program may be archived on HD for playback or for higher level interpretation and display done by the Science Display Module (SDM) on a different computer.
The paragraphs 3.1 describes in detail the operation at ILT, only the difference at SLT will be described in paragraph 3.2.

3.1 Instrument Level Test Operations

3.1.1 Program setup

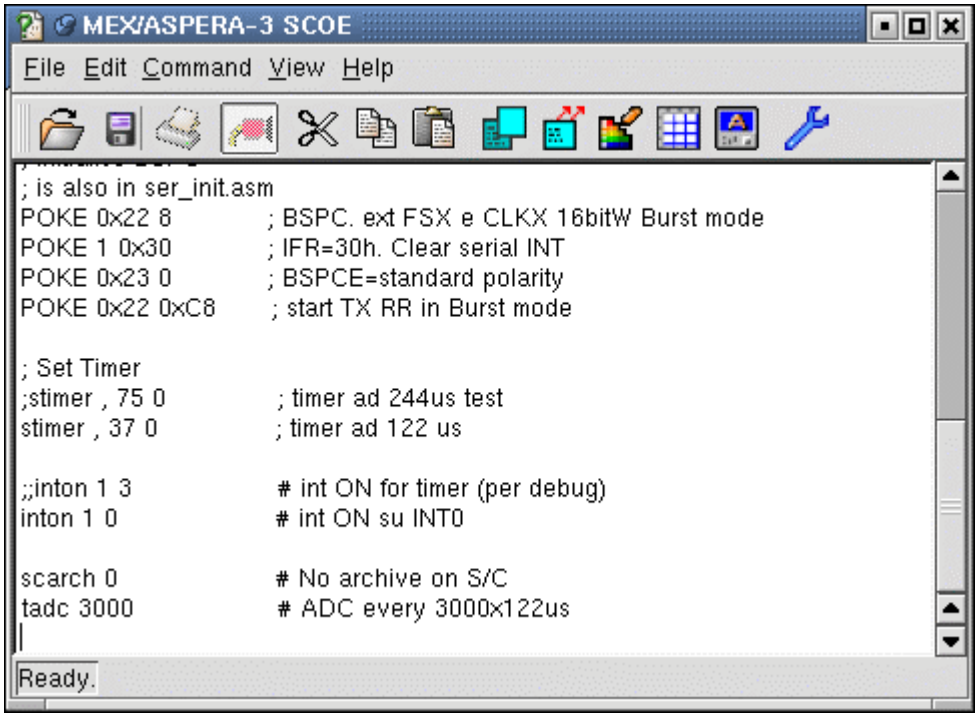
The SCOE programs used at ILT are **scoe** and **mexilt**. Both programs must be executed manually or via the **aspera** script procedure. Links to the programs are in /Mex directory. For program configuration see paragraph 2.4.

3.1.2 SCOE interface

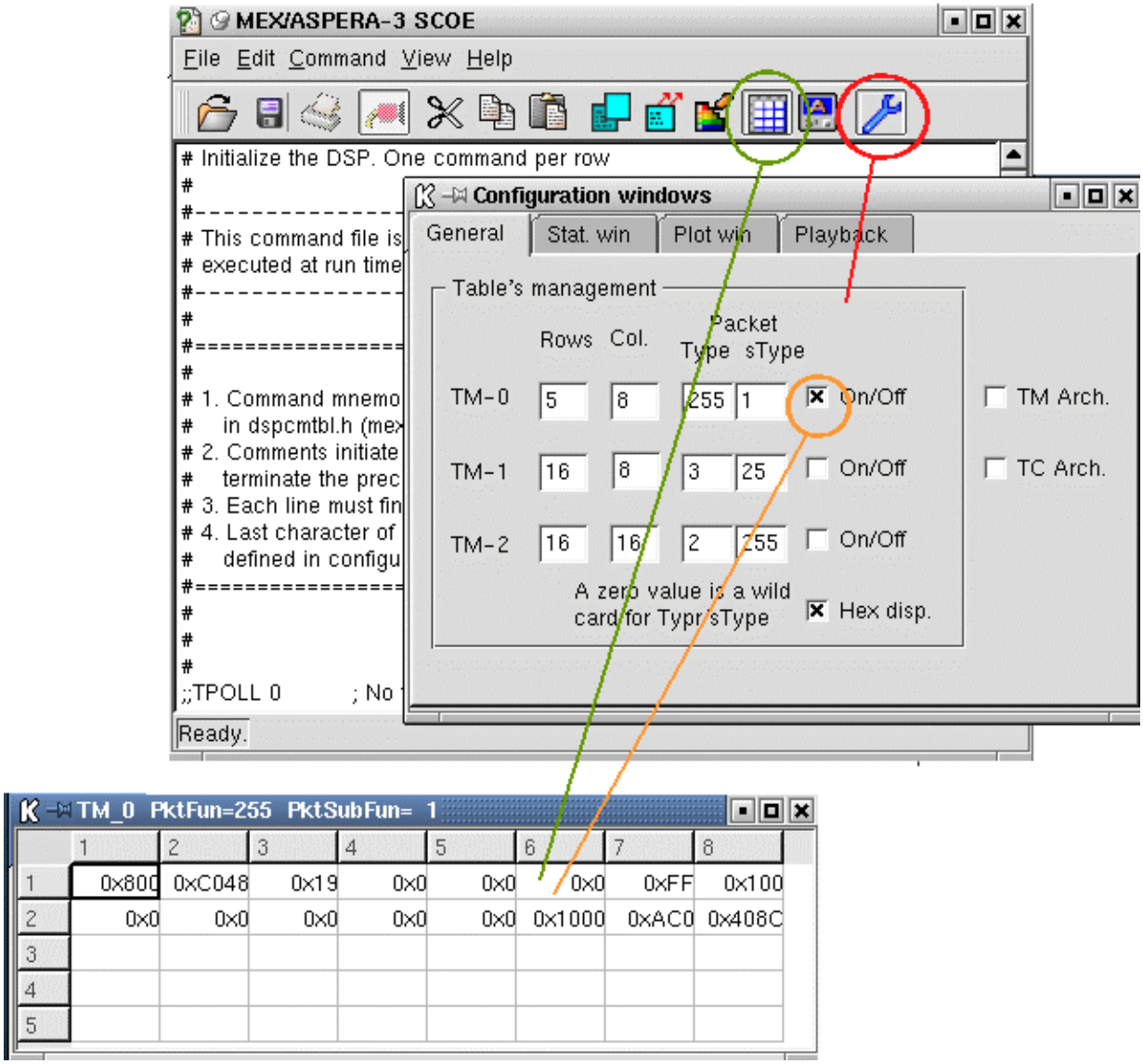
The main EGSE interface is the **scoe** program. The program consists of an editor-like window used to compose, edit and send commands to ASPERA via **mexilt** and **S/C sim**.
Each line in the editor window is a single case-insensitive command or a comment (beginning with '**#**' or '**;**' character). Text after '**#**' or '**;**' characters is ignored.
The text present on the window may be stored/retrieved to/from a file.
Commands lines can be sent one by one, or as a sequence at a defined time interval.
The following function-keys have a special meaning in the editor window:

- The **F9** (or **Ctrl-Return**) key sends to **mexilt** the current (where the cursor is) command line or the block of lines selected with the mouse.
- The **F11** key sends to **mexilt** the current command line and moves the cursor to the next line, empty and comment lines are ignored.
- The **F10** key sends to **mexilt** all the commands from the current line as a procedure (see the following "commands sequences" paragraph).

The figure shows the **scoe** main window with the set of initialisation commands executed automatically at startup


MEX/ASPERA-3 SCOE
File Edit Command View Help
; is also in ser_init.asm
POKE 0x22 8 ; BSPC. ext FSX e CLKX 16bitW Burst mode
POKE 1 0x30 ; IFR=30h. Clear serial INT
POKE 0x23 0 ; BSPCE=standard polarity
POKE 0x22 0xC8 ; start TX RR in Burst mode
; Set Timer
;stimer , 75 0 ; timer ad 244us test
stimer , 37 0 ; timer ad 122 us
;;inton 1 3 # int ON for timer (per debug)
inton 1 0 # int ON su INT0
scarch 0 # No archive on S/C
tadc 3000 # ADC every 3000x122us
Ready.

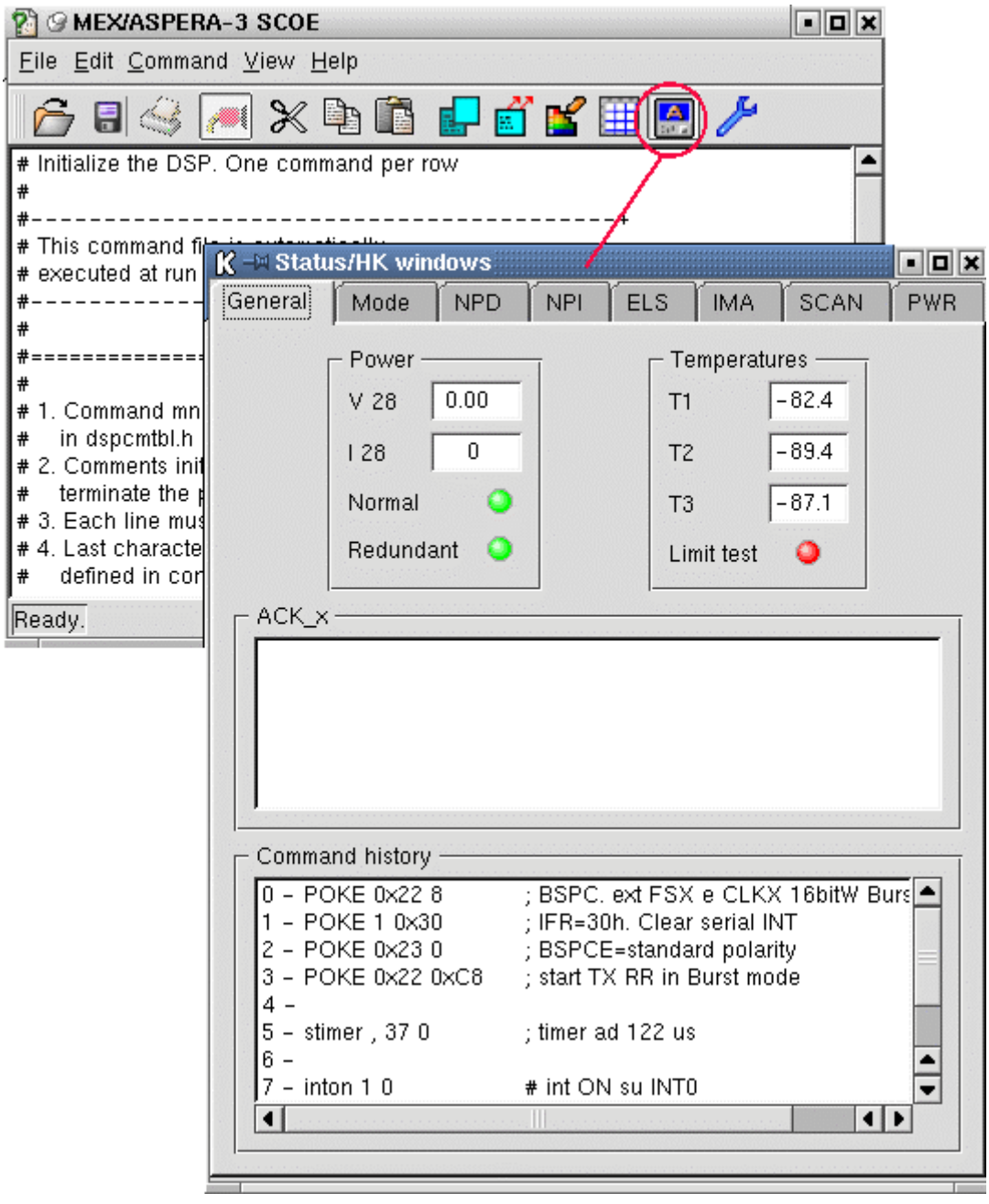
The main configuration (modless) window, shown in the following figure, enables tabular windows view of selected telemetry packets (up to 3 windows) and the archiving of all the telemetry packets and/or commands.
Telemetry packet type=255 is reserved for private S/C simulator – scoe housekeeping and is archived in a different file.



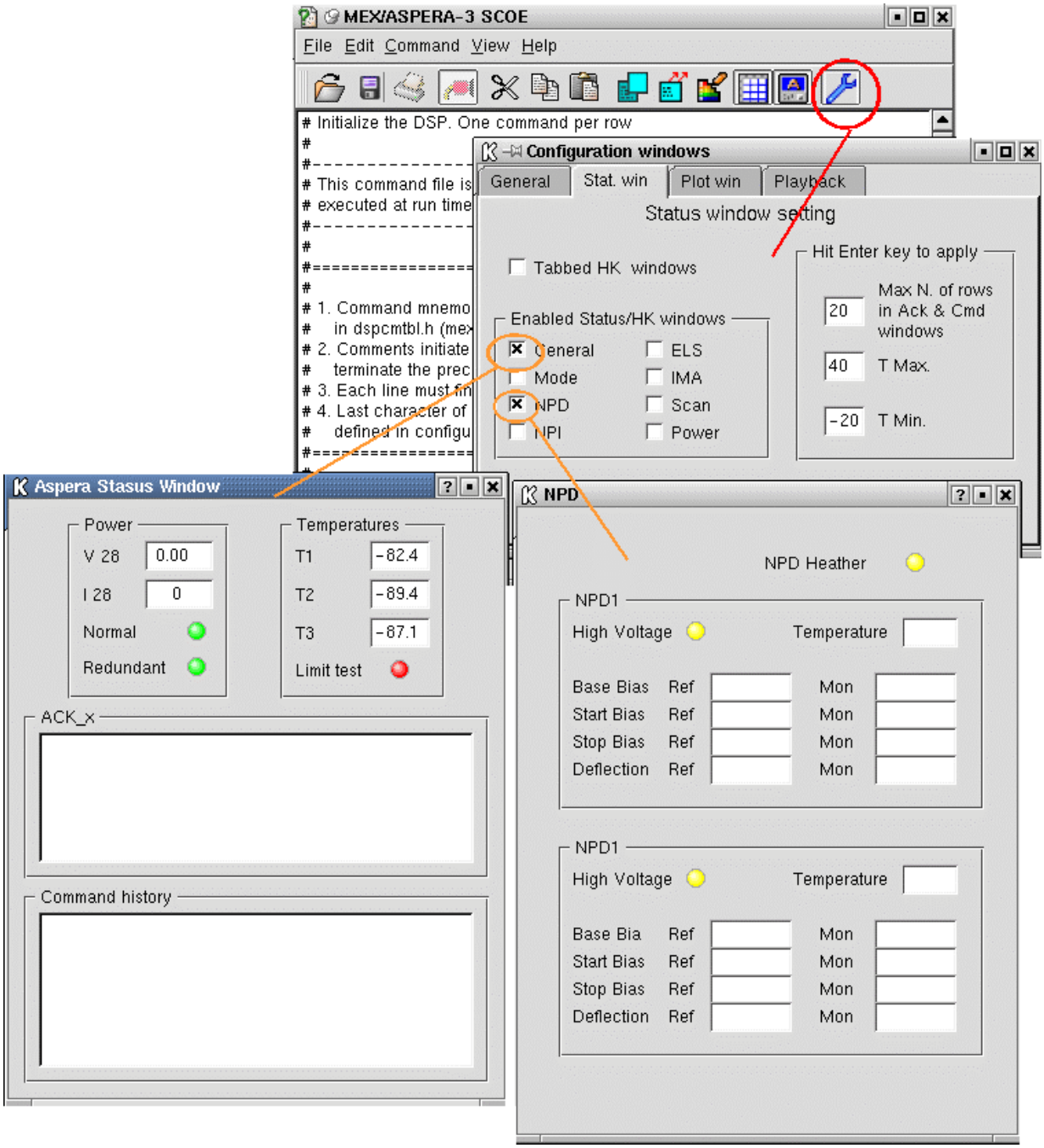
The display can be in dec or hex notation. A zero function/subfunction match all function/subfunction. Archive files are stored in /Mex/Archive directory, and the files name indicate the creation time. A new file is generated at fixed time as defined on /Mex/scoerc configuration file.

Up to 4 selected parameters may be plotted in a strip_chart windows (see section 3.1.6.3)

The next figure shows the multi-tabbed modless status/housekeeping window and the associated command history frame. The full history can be stored in the TC archive file.



The ASPERA-3 housekeeping, transmitted as packet (3,25), is displayed in the tabbed windows. The 8 (Status/HK windows) may be also selectively displayed as separate windows as shown in the next figure



3.1.3 Command List

As a general rule all commands are case insensitive
Commands instructions are coded as a single text line beginning with the command ID (acronym) and followed by parameters (if any):

```
ID [p1 p2  …  pn]
```

ID and parameters are separated by 'space', 'tab' or ',' (comma separator is **not allowed** in commands at system level).
Characters following ';' or '#' are comments.
Lines of comments beginning with '#' in column one, are saved in the command archive file.
Hexadecimal constants use the C coding convention (i.e. 0x5A5A).

There are five types of commands each type with its own definition file:

- **SCOE internal**: These commands are used to set scoe program flags/variable (which remain constant up to the next setting). The commands are defined in the first section of */Mex/include/dspcmtbl.h* and mustn't be modified.

- **Low level**: these commands are sent directly to the mexilt module and operate at the lowest level. The commands are defined in second section of */Mex/include/dspcmtbl.h* and mustn't be modified.

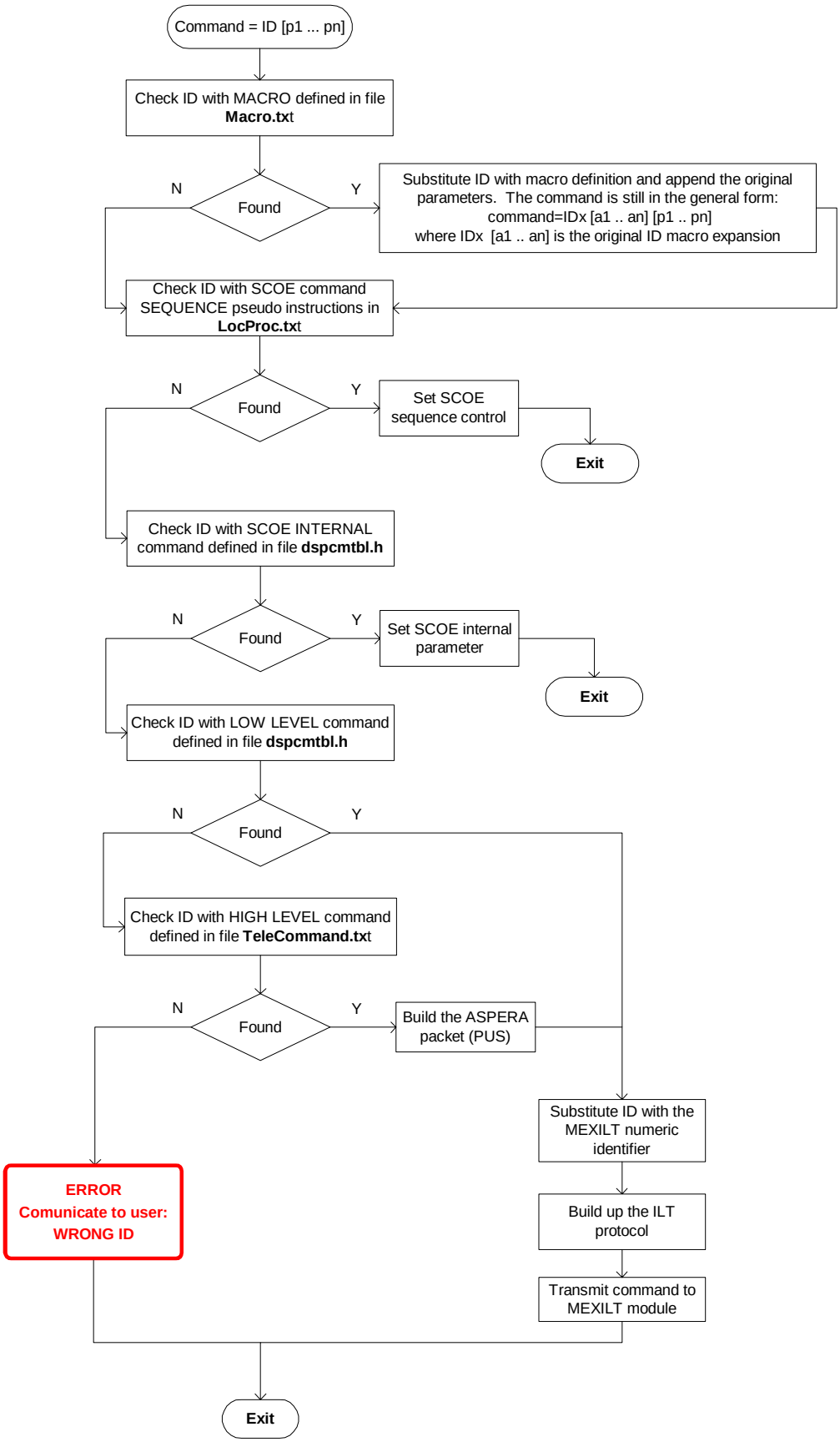
- **High level**: these commands are composed by an acronym possibly followed by parameters and generate a standard packet telecommand. The commands are translated in the scoe module to low level commands and transmitted to mexilt. The commands are defined in */Mex/CmdLut/TeleCommand.txt* and are intended to be modified by the user.

- **Macro**: these commands just substitute the macro acronym with whatever follows in the macro definition line. A macro acronym may be followed by parameters in that case the parameters are appended at the end of the macro translation line. The commands are defined in */Mex/CmdLut/Macro.txt* and are intended to be generated/modified by the user.

- **Command sequence instructions**: these are pseudo programming-language instructions that control the flow of commands in the sequences. The commands are defined in */Mex/CmdLut/LocProc.txt* and in */Mex/include/dspcmtbl.h* and mustn't be modified.

The following flow diagram shows the command verification and implementation algorithm

Command decoding diagram


Command = ID [p1 ... pn]
Check ID with MACRO defined in file Macro.txt
N
Found
Y
Substitute ID with macro definition and append the original parameters. The command is still in the general form: command=IDx [a1 .. an] [p1 .. pn] where IDx [a1 .. an] is the original ID macro expansion
Check ID with SCOE command SEQUENCE pseudo instructions in LocProc.txt
N
Found
Y
Set SCOE sequence control
Exit
Check ID with SCOE INTERNAL command defined in file dspcmtbl.h
N
Found
Y
Set SCOE internal parameter
Exit
Check ID with LOW LEVEL command defined in file dspcmtbl.h
N
Found
Y
Check ID with HIGH LEVEL command defined in file TeleCommand.txt
N
Found
Y
Build the ASPERA packet (PUS)
Substitute ID with the MEXILT numeric identifier
Build up the ILT protocol
Transmit command to MEXILT module
ERROR
Comunicate to user:
WRONG ID
Exit

3.1.3.1 SCOE internal

This first block of commands is used internally to the SCOE program. These commands are used to set internal flags/variable (which remain constant up to the next setting) or check the telemetry data (VERTM).

ID	N. Par	Param.	Description
IMME	0		Follows immediate cmd
PROG	0		Follows program cmd
NORM	0		Follows normal cmd
ASAP	0		Same as NORM. For back-compatibility
EXP	0		Follows Experiment request commands. This is normally an ASPERA request (event)
HEX	0		Follows parameters in Hex format (no 0x.. prefix requested)
DEC	0		Follows parameters in Dec format
ACKA	1	Flg=0/1	Ack report on/off on acceptance of pkt
ACKE	1	Flg=0/1	Ack report on/off on execution of pkt
APID	1	61 - 65	App ID for pkt
NPAC	1	N	Set to N the # pkt for current APID (it is auto incremented)
INFO	1	N	Set to N the SCOE local info (echo) level.
STATE	1	N	Set to N the SCOE state machine level
VERHK	6	TO, Sid, Offs, Msk, Mn, Mx	**This command is only valid in a procedure.** Halt scoe local procedure (with time out TO in seconds) untill HK word do match the following parameters: TimeOut, SID, SourceDataOffset, Mask, Min, Max. Offset is referred to SID position in HK packet (i.e. 8 words from beginning of packet. *ONLY FOR BACK COMPATIBILITY. Use VERTM*

VERTM	n	Offs, Msk,Val …….. Offs, Msk,Val Offs, Msk, Min, Max	**This command is only valid in a procedure.** Halt scoe local procedure (with time out TO in seconds defined by RX[15]) until TM words do match the following parameters: [Offset,Mask,Value] up to 8 times, Offset, Mask,Min, Max If Time-Out occurs, jump RX [14] lines. Here offset is counted from the beginning of the TM packet. Refer to paragraph 3.1.3.5.1
PLOT	n	Np, Offs, Msk,Val …….. Offs, Msk,Val Offsx, Mskx, Min, Max	This command set the strip_chart plot Number Np**.** The program look in telemetry until TM words do match the following parameters: [Offset,Mask,Value] up to 8 times, then send to the strip_chart plotter the word defined by: Offsx, Mskx, Setting the plotting limit to Min, Max See section 3.1.3.6
TXFILE[•]	1	Filnam	Transmit the named command file. The file must be in /Mex/CmdFiles/

[•] A command file executed via the TXFILE command, can't include procedure "pseudo instructions"

3.1.3.2 Low level commands

The following tables show the low level commands recognized by MEXILT.
The commands in bold type are those normally used to test ASPERA while the other commands are mainly intended for internal debugging.
The low level commands are coded in */Mex/include/dspcmtbl.h* and mustn't be modified by the users

This block of commands, executed inside the MEXILT program, is special S/C command.

ID	N. Par	Param.	Description
HALT	0		Terminate application
RESET	1	N (Bitmap)	(N=xx xxx1b)=> Reset circ. buff I (N =xx xx1xb)=> Reset circ. buff P (N =xx x1xxb)=> Reset circ. buff N (N =xx 1xxxb)=> retransmit API (test) (N =x1 xxxxb)=> close the socket (N =1x xxxxb)=> reset TM expected words
ECHO	1	0/1/2	Request different level of echo (default 1)
RUNP	1	# prog	Run prog. N. [1 .. x]
ENDP	0		Terminate current program
SETRX	1	+-RX	Increment decrement RX. Inc[bit4=0], dec[bit4=1]. RX[bit 0..3]
	2	RX, N	Store N in RX. RX[bit 0..3]
MOVRX	2	RXn,RXm	Copy RXn to RXm.
READRX	2	RXn,nRX	Transmit nRX. Starting from RXn
JUMP	1	RX+-N	Jmp +-N w16. If RX==0 jump unconditioned. Rx[bit 15..12], num[bit 9..0] (-num [bit 10=1]). If bit 11=1 decrement Rx (after jump). If Rx!=0 => jump until Rx!=0. Es: jump 2c08(Hex) => jump -8 tlc until Rx[2]>0 and decrement Rx[2] NOTE that each command is formed by: Acronym, Nw16, and [parameters]. So to skip back the only JUMP instruction you have to skip back 3 w16
WAIT	1	Nsec	Sleep for N seconds
WAITRX	1	RX	Sleep for RX milliseconds
WAITMS	1	Nmsec	Sleep for N msec milliseconds
WRITE	x	Add,w1,..	Write in mexilt memory, starting from ADD, the following words w1 w2 w3 …
READ	2	Add,Nw	Transmit N word16 beginning from Add
CBREP	0/1	N	Circular Buffers Report. Send back, as pkt Type=255 sType=0, the first N mexilt circ. buffer free area length. Buffer sequence is: Immediate cmd buffer Program buffer Normal cmd buffer Instrument cmd buffer

			5 DSP internal buffer Default N = 3
TPOLL	1	Nsec	Set TM polling every Nsec. If 0 no polling
SCSEND	1	0/1	Send to scoe telemetry sent only to dms. Default 0 (do not send)
SCARCH	1	0/1	Archive dms telemetry. Default 1 (Archive)
TXCTX	1	apid	Send the Context to ASPERA (apid)

The following commands are sent to the DSP board. Code here MUST match those in the DSP program (MEXGSE).

ID	N. Par	Param.	Description
RST	0		Reset DSP pointers
POKE	n	Start Add, P1, P2, ..	Write on DSP memory starting from “Start Add” P1, P2 …
PEEK	2	Start Add, N	Read from DSP memory starting from “Start Add” N words and transmit in TL as Pkt Type=255, sType=0
FUN4	1		Utility
STIMER	2	R1, R2	Set Timer. 2 parameters: R1,R2
INTON	2	rgN,Dat	On Off interrupt N
TLC	n	rgN	Send n TC
TLM	1	Nw	Ask for Nw TM words and transmit to SCOE as Pkt Type=255, sType=2
TLMRX	1	Rx	Ask for Nw=*Rx TM words
TMON	1	0/1	Suspend/restart current Telemetry request
ADREG	1	Reg	Modify a AC01 Register (MSByte= reg num)
ADC	0		Read in ADBUFS all ADC channels Inhibit interrupt (SL) for .5ms. To get value use peek at ADBUFS
TADC	1	T	Read the 8 ADC channels every t *122us and transmit to SCOE as Pkt Type=255, sType=1
RELAY	2	RN ,t	Pulse to relay rN [0..3] for t time in unit of 122 us. Relay # 0 => Main 28V On “ “ 1 => Main 28V Off “ “ 2 => Redundant 28V On “ “ 3 => Redundant 28V Off
STIM	0		Send to ASPERA Time Packet. Time may be set with poke at TIMBUFS.
PKT	n	Type, Subtype, Param…	Send Packet TC

The most important commands in this block are the **TPOLL** and **PKT** command. The following commands block sends packet Type 3, sub type 5, SID 10 (enable HK packet generation) to the instrument APID=61 requesting acceptance acknowledge without execution acknowledge and initiate the telemetry polling every 2 seconds:

```
acka  1        ; until notified by a new ACKA cmd, all commands
               ; require acceptance acknowledge
acke  0        ; until notified by a new ACKE cmd, execution
               ; acknowledge is not requested
pkt   3  5  10 ; sends packet Type 3, sub type 5, SID 10 (enable
               ; HK packet generation)
```

```
tpoll 2        ; Start polling for telemetry every 2 seconds
```

3.1.3.3 High Level Commands

These commands are formed by an acronym possibly followed by parameters and generate a standard packet telecommand. The commands are translated in the scoe module to low level commands and transmitted to mexilt.
The tables in appendix 6.2, shows the ASPERA specific commands. APID, Packet Type and subtype are shown here only for internal documentation.
The first 3 columns of this table are recorded in the file TeleCommand.txt used as a lookup table by SCOE for high level commands decoding and are intended to be modified by the user.

File location: /Mex/CmdFiles/*TeleCommand.txt*

As an example the command:

```
ASPMELS30
```

send to the instrument APID=61 a telecommand packet type=191 subtype=1

3.1.3.4 Macro Commands

These commands just substitute the macro acronym with whatever follows in the macro definition line. A macro acronym may be followed by parameters in that case the parameters are appended at the end of the macro translation line.
The macro expansion may refer to any of the low or high-level commands and command sequence instructions.

The commands are defined in */Mex/CmdLut/Macro.txt* and are intended to be generated/modified by the user.
Here are the initial instructions for the macro definition file

Macro name	Macro expansion	Comments
MAIN28on	relay 0 200	; Pulse relay # 0 for 200x122us = 24ms
MAIN28off	relay 1 200	
RED28on	relay 2 200	
RED28off	relay 3 200	
MAINCURRENT	`PLOT 0 6 0xff 0xff 7 0xff00 0x100 12 0xffff 0 0x7fff`	Set the plot # 0 (red) to show the main bus current
_TOVer	_SETRX 15	; Set time-out for VERTM
_TOJmp	_SETRX 14	; Set jmp on time-out on VERTM

3.1.3.5 Commands sequences

A command sequence is formed by standard commands plus a few "pseudo instructions" special commands, defining elementary programming language statements (i.e. for loop, if statements, setting of program variables etc).

At Instrument Level Test (ILT) configuration, two kind of command sequence are possible: procedure and program. The sequences differ in the execution location (SCOE/MEXILT) and in the pseudo instruction mnemonic.

3.1.3.5.1 Procedure

A procedure is a sequence of commands, as defined in the previous sections, stored on the SCOE main window. As such each command is transmitted to MEXILT every time it must be executed.
The procedure is controlled by few "pseudo instructions":

Instruction	Parameters	Description
_STEP	N	N=Time [ms] to wait for the next instructions until redefined. Default N=200
_SETRX	Rn, X	Set local register R[r](r= 0 .. 15)to X. R[r]=X
_RX+	r, +-X	R[r]= R[r]+X (X may be negative)
_RX-RY	r , n	R[r]= R[r]- R[n]
_RX+RY	r , n	R[r]= R[r]+ R[n]
_JMP	+-N	Skip +-N lines (comments and empty lines ignored). _JMP –1 execute the line before the _JMP _JMP 0 execute the line after the _JMP. This is an exception to avoid dead loop _JMP 1 execute the line after the _JMP
_JMPRX	r ,+-N	Skip +-N lines (comments and empty lines ignored) and decrement R[r]. If R[r]=0 No skip
_GOTO	_str_	Goto to label _str_. Labels must begin/end with '_' and are case insensitive.
_GOTORX	r, _str_	Goto to label _str_ and decrement R[r]. If R[r]=0 no jump
_BEGIN		Initiate the procedure. Used only to zeros the errors counter
_END		Terminate the procedure and print total errors

The procedure initiate by the function key **F10** at the current cursor position, and terminate when the _END instruction is reached, no more instructions are present on the edit window/command file or pressing again the **F10** key.
The procedure "pseudo instructions" are strongly connected to the editor window layout, so these instructions can’t be used inside a command file transmitted directly as with TXFILE command.
The array RX[0 .. 15] is an integer general-purpose variable array to be used inside the procedures. The last 4 variables have a special meaning:

- RX[12] TBD
- RX[13] TBD
- RX[14] Used as jump RX[14] lines if VERTM command end with time-out (predefined RX[14]=1).
- RX[15] Used as time-out (seconds) in VERTM command (predefined RX[15]=20).

A typical procedure could be:

```
_begin                  ; Initialise errors counter
_step 2000              ; 2 seconds time interval between instructions
MAIN28on                ; Switch on main 28 V. This macro expand to
                        ; relay 0 200 which send a 15V pulse to the
                        ; relay # 0 with a duration of 200x122us = 24ms
tpoll 3                 ; poll for telemetry every 3 seconds
verhk  10 0  6 0xff00 0x1000 0xff00
                        ; wait (with a timeout of 10 seconds) for Hk SID=0
                        ; offset 6 words (16 bit word), byte high to be
                        ; greater than 16
_tover  10              ; This predefined macro expand to _SETRX 15 10 and
                        ; set time-out for all subsequent VERTM command
                        ; to 10 seconds
_tojmp  2               ; This predefined macro expand to _SETRX 14 2 and
                        ; set the  jump linees on time-out for all subsequent
                        ; VERTM command to 2

vertm  6 0xff 3 7 0xff00 0x1900 8 0xff 0 14 0xff00 0x1000 0xff00
                        ; This command is equivalent to the previous (actually
                        ; VERHK is translated to VERTM) but much more general.
                        ; The command select the HK packet [Type 3 (6,0xff,3)
                        ; subType 25 (7,0xff00,0x1900)] sid=0 (8,0xff,0) then
                        ; check the same parameter which is now at offset
                        ; 6+8=14 because in VERTM the offset is counted from
                        ; the beginning of the packet.

_jmp 2                  ; jump to _SETRX 0 3 instruction if no timeout
                        ; occurred in the previous vertm

_goto  _error_          ; go to label _error_ if timeout occurred in the
                        ; previous vertm

_SETRX 0 3              ; set local register #0 to 3
ASPMELS30               ; send the high level command to Aspera as listed in
                        ; table /Mex/CmdLut/TeleCommand.txt

_tover  30              ; This predefined macro expand to _SETRX 15 30 and
_tojmp  1               ; This predefined macro expand to _SETRX 14 1

vertm  6,0xff,3, 7,0xff00,0x1900, 8,0xff,0, 17, 0xff0f, 0x1000, 0xff00
                        ; verify command execution on HK
                        ; Sid=0,timeout 30 sec ..........
ASPMNPD1Start
vertm  6,0xff,3, 7,0xff00,0x1900, 8,0xff,0, 27, 0xff0f, 0x1000, 0xff00
                        ; verify command execution on HK
                        ; Sid=0, timeout 30 sec ..........
ASPMNPD1Stop
verhk 0, 29 0xff0f, 0x1000, 0xff00
                        ; verify command execution on HK
                        ; Sid=0, timeout 30 sec ..........
```

```
_JMPRX 0,-8            ; go back to command ASPMELS30 for 3 times
tpoll 0                ; stop telemetry polling
MAIN28off              ; switch off main 28V power line
…
_error_                ; Begin of error section
…
```

Comments:

- Use VERTM instead of VERHK, the latter is shown only for back compatibility and is internally translated to VERTM. For frequent use of VERHK, write a macro command like:

  ```
  TestHkSid  vertm  6 0xff 3  7 0xff00 0x1900  8 0xff
  ```

  so that the previous instruction could be written like:

  ```
  TestHkSid 0 14 0xff0  0x1000 0xff00
  ```

  just remember to set the time-out with _TOVER 20

- Try to avoid commas as separator, this is legal at instrument level test but is not permitted at system level tes when connectet to the CCS. Use spaces.

3.1.3.5.2 Program

A program is a sequence of low level commands (3.1.3.1) transmitted from SCOE to MEXILT, stored and executed on MEXILT. MEXILT store the programs in a 512 16 bit word circular buffer so that more programs can be stored, each one numbered in sequence.
The following table shows the "pseudo instructions" available for programs.

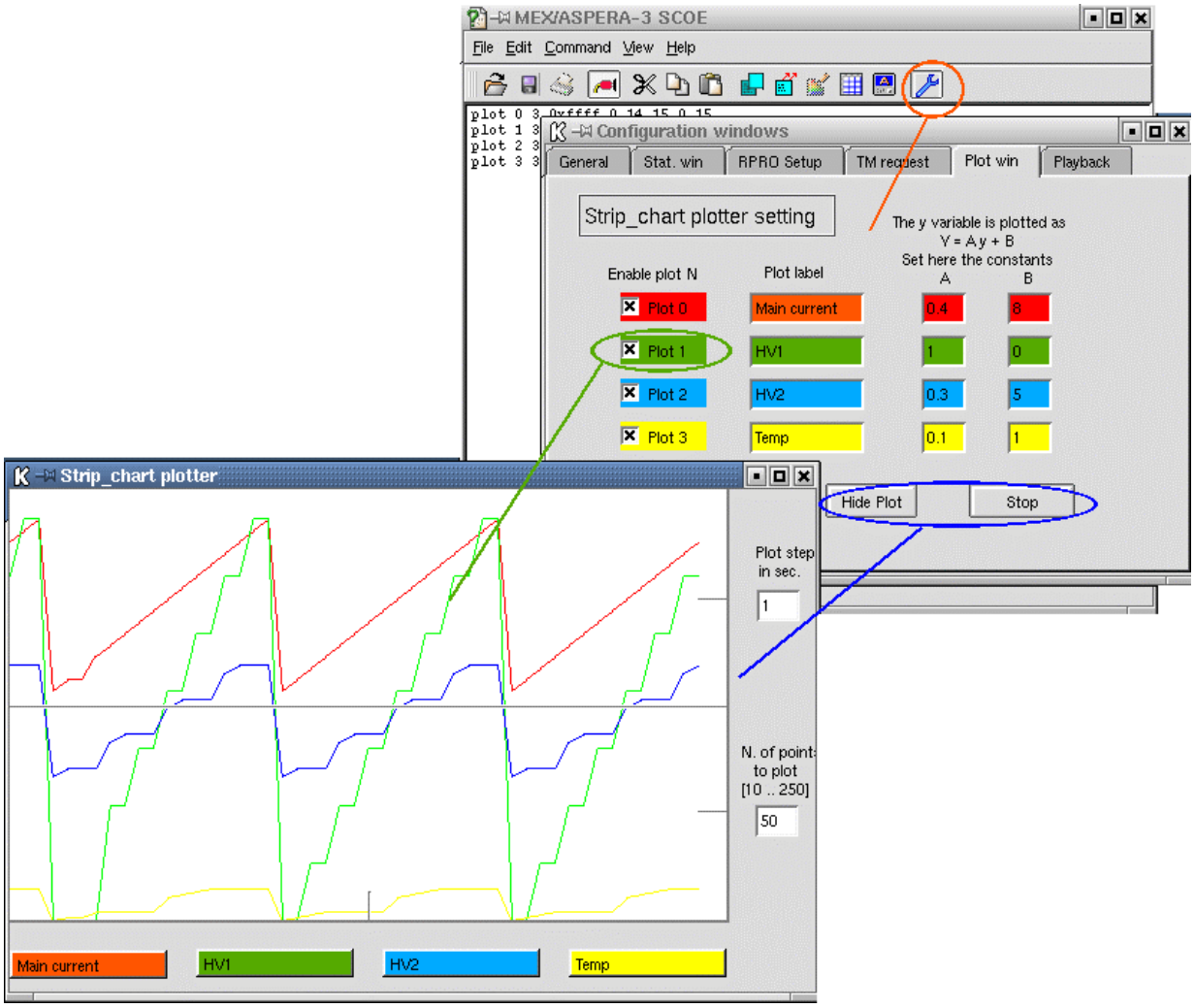
Instruction	Parameters	Description
HALT		Terminate application
RUNP	# prog	Run prog. N. [1 .. x]
ENDP		Terminate current program
SETRX	+-RX	Increment decrement RX. Inc[bit4=0], dec[bit4=1]. RX[bit 0..3]
	RX, N	Store N in RX. RX[bit 0..3]
MOVRX	RXn,RXm	Copy RXn to RXm.
READRX	RXn,nRX	Transmit nRX. Starting from RXn
JUMP	RX+-N	Jmp +-N w16. If RX==0 jump unconditioned. Rx[bit 15..12], num[bit 9..0] (-num [bit 10=1]). If bit 11=1 decrement Rx (after jump). If Rx!=0 => jump until Rx!=0. Es: jump 2c08(Hex) => jump -8 tlc until Rx[2]>0 and decrement Rx[2] NOTE that each command is formed by: Acronym, Nw16, and [parameters]. So to skip back the only JUMP instruction you have to skip back 3 w16
WAIT	Nsec	Sleep for N seconds
WAITRX	RX	Sleep for RX milliseconds
WAITMS	Nmsec	Sleep for N msec milliseconds

In order to transmit a program to MEXILT the command sequence must be preceded by the **PROG** low-level command. At the end of the program transmission use the **IMME** or **NORM** low-level command
Example:

```
Prog              ; Set program mode

tlc 0xaa55        ; send telecommand aa55 hex
wait 3            ; wait 3 seconds
tlc 0x5aa5        ; send telecommand 5aa5 hex
wait 3            ; wait 3 seconds
jump 0x40f        ; jump –15 to "tlc 0xaa55"
endp              ; end of program (useless in this example)

imme              ; Set immediate mode

runp 1            ; execute the first stored program.
                  ; To terminate send ENDP as IMME command
```

3.1.3.6 Strip_chart Plotter

The strip_chart plotter is enabled as shown in the next figure



The plotted parameters (up to 4) must be defined with the PLOT SCOE internal command.

As an example to plot the main bus current as plot # 0:

```
PLOT  0  6 0xff 0xff  7 0xff00 0x100  12 0xffff 0 0x7fff
```

which define:

0 Plotted line # 0 (red)

6 offset 6 words
0xff Mask for packet type
0xff expected packet type is 0xff (used at ILT for internal HK)

7 offset 7 words

0xff00 Mask for packet subtype
0x100 Expected subtype is 1

12 Offset 12 words (is the current monitor)
0xffff Keep the full word (an plot it)
0 Plot lower limit
0x8000 Plot upper limit

This instruction is inserted in the macro definition file as MAINCURRENT.

3.1.4 Archiving

The appropriate switch in the configuration window tab 1 ("General") enables data archiving.

TM Archive: this switch enables the binary archiving of all telemetry packets received by the scoe. Archiving is performed in files:

/Mex/Archive/Myymmdd_hhmmss.tlm
with: yy =year
mm = month
dd = day
hh = hour

The telemetry archive is closed and reopened (with a new name) at time interval defined in scoerc (see 2.2) and resynchronized at integer unit of the time interval

```
…
    #-------- Seconds to reopen TM archive
SEC_TO_NEW_ARCTM=3600
…
```

The engineering S/C sim data (Pkt 255,x) are archived in a different file:

/Mex/Archive/Myymmdd_hhmmss.tl.Loc

using the same naming convention.

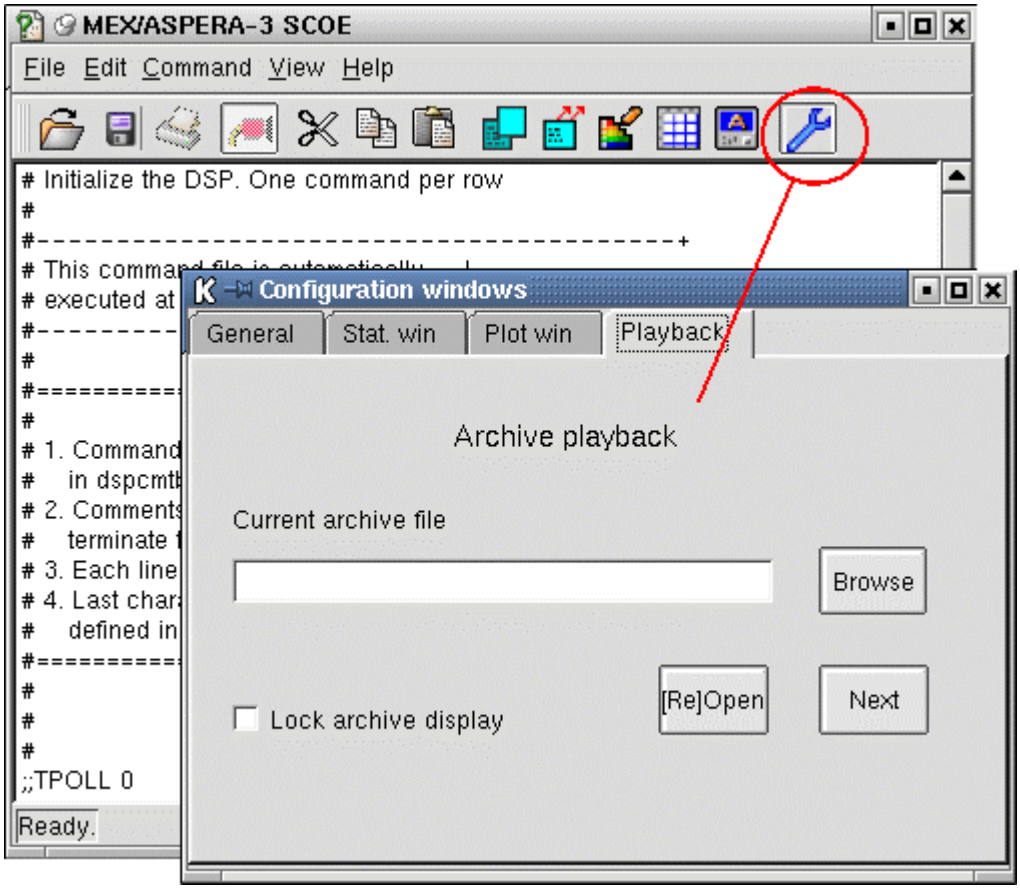
TC Archive: this switch enables the ASCII archive of commands sent by scoe to S/C sim. The commands are stored in file

/Mex/Archive/Cyymmdd_hhmmss.cmd

using the same naming convention as for TM archive. The commands are stored with the actual time.

Lines of comments beginning with '#' in column one, are saved in the command archive file.

3.1.5 Playback



3.2 System Level Test Operations

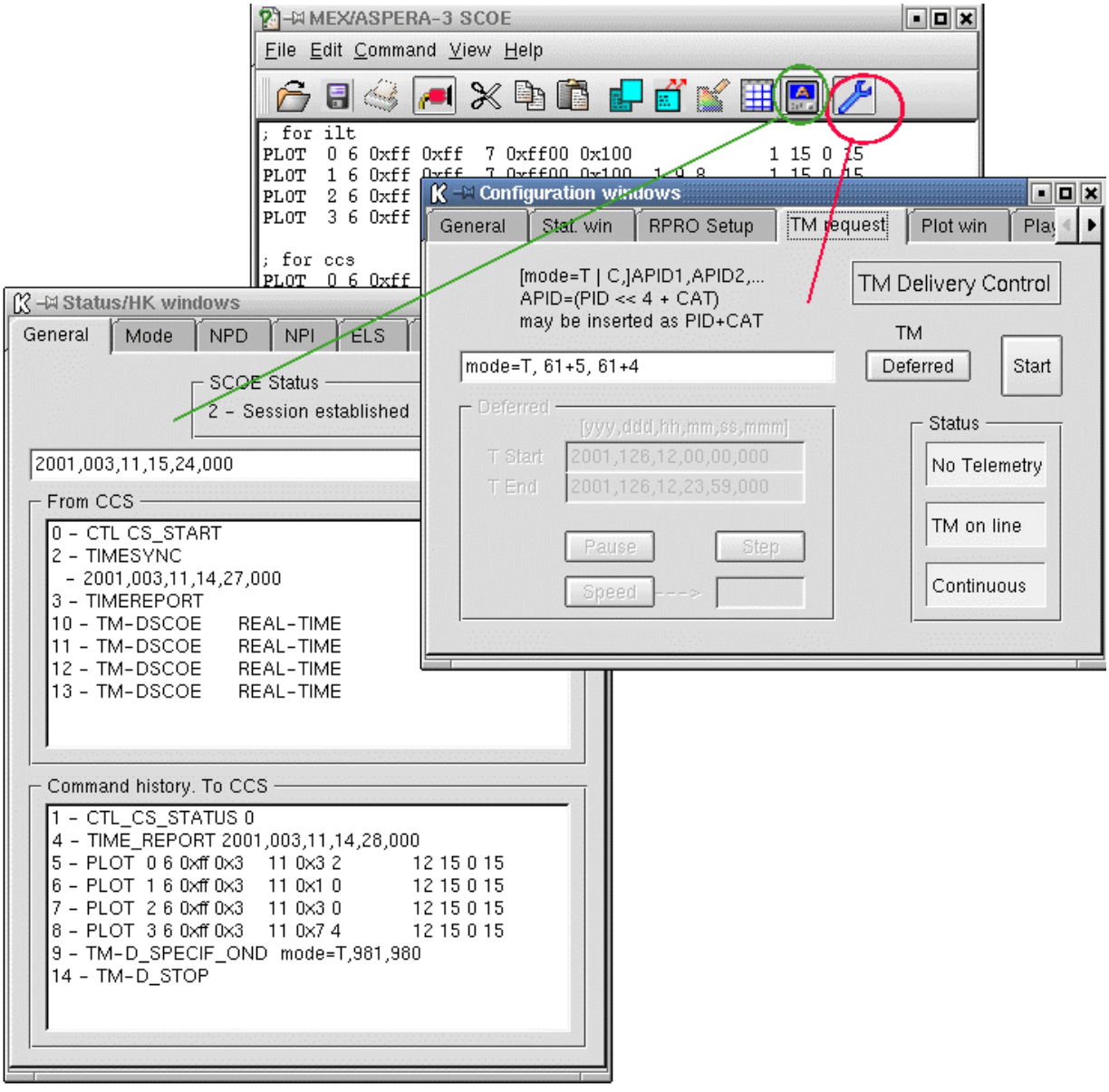
The functionalities of the **scoe** program at System Level Test (SLT) are essentially the same as for ILT (sect 3.1). In this sections only the differences with ILT will be described.

3.2.1 Program Setup

Only the **scoe** program is used at SLT. The program must be recompiled as explained in sect. 2.5 and executed manually. Links to the program and associated files are in /Mex directory.

3.2.2 SCOE Interface

The interface is the same of the ILT with a few differences outlined in the following figure:



- The configuration-tabbed dialog shows two extra windows, the most important is the TM request used to manually request the telemetry packets to the CCS.
- The first window in the Status/HK tabbed dialog is slightly different and shows the full history of commands flow from/to CCS. In the figure is shown the initial automatic command exchange used by CCS to initiate the connection to the SCOE:

CCS send	***SCOE Reply***
CTL CS_START	
	CTL CS_STATUS 0
TIMESYNC 2000,362 ….. TIMEREPORT	
	TIME REPORT 2000,362 ….

3.2.3 Command List

Here again there are few differences with ILT:

- The ‘,’ is not allowed as separator in command parameters as the comma character is used in the CCS parameters strings as separator. Never use a space separator in CCS parameters string.
- There are no Low Level commands as the mexilt program is not used.
- A new set of CCS Protocol Commands is implemented.
- Only few of SCOE internal commands are meaningful:

There are five types of commands each type with its own definition file:

- **SCOE internal**: these commands are used to set scoe program flags/variable (which remain constant up to the next setting) . The commands are defined in the first section of */Mex/include/dspcmtbl.h* and mustn’t be modified.

- **ASPERA telecommand**: these commands are composed by an acronym possibly followed by parameters and generate a standard RPRO telecommand request (TC-R SYM_TC). The commands are defined in */Mex/CmdLut/TeleCommandSLT.txt* and mustn’t be modified.

- **CCS protocol command**: these commands are composed by an acronym possibly followed by parameters and generate a standard RPRO packet command. The commands are defined in */Mex/CmdLut/Rpropkt.txt* and are intended to be modified by the user.

- **Macro**: these commands just substitute the macro acronym with whatever follows in the macro definition line. A macro acronym may be followed by parameters in that case the parameters are appended at the end of the macro translation line. The commands are defined in */Mex/CmdLut/MacroSLT.txt* and are intended to be generated/modified by the user.

- **Command sequence instructions**: these are pseudo programming-language instructions that control the flow of commands in the sequences. . The commands are defined in */Mex/CmdLut/LocProc.txt* and mustn’t be modified.

3.2.3.1 SCOE internal

This first block of commands is used internally to the SCOE program. These commands are used to set internal flags/variable (which remain constant up to the next setting) or check the telemetry data (VERTM).

ID	N. Par	Param.	Description
INFO	1	N	Set to N the SCOE local info (echo) level.
STATE	1	N	Set to N the SCOE state machine level
VERHK	6	TO, Sid, Offs, Msk, Mn, Mx	**This command is only valid in a procedure.** Halt scoe local procedure (with time out TO in seconds) until HK word do match the following parameters: TimeOut, SID, SourceDataOffset, Mask, Min, Max. Offset is referred to SID position in HK packet (i.e. 8 words fron beginning of packet. *ONLY FOR BACK COMPATIBILITY. Use VERTM*
VERTM	n	Offs, Msk,Val …….. Offs, Msk,Val Offs, Msk, Min, Max	**This command is only valid in a procedure.** Halt scoe local procedure (with time out TO in seconds defined by RX[15]) until TM words do match the following parameters: [Offset,Mask,Value] up to 8 times, Offset, Mask,Min, Max If Time-Out occur, jump RX[14] lines. Here offset is counted from the beginning of the TM packet. Refer to paragraph 3.1.3.5.1
PLOT	n	Np, Offs, Msk,Val …….. Offs, Msk,Val Offsx, Mskx, Min, Max	This command set the strip_chart plot Number Np**.** The program look in telemetry until TM words do match the following parameters: [Offset,Mask,Value] up to 8 times, then send to the strip_chart plotter the word defined by: Offsx, Mskx, Setting the plotting limit to Min, Max See section 3.1.3.6
TXFILE[•]	1	Filnam	Transmit the named command file. The file must be in /Mex/CmdFiles/

[•] A command file executed via the TXFILE command, can't include procedure "pseudo instructions"

3.2.3.2 ASPERA telecommand

These commands execute a CCS Telecommand Request [RD1] using a private ASPERA acronym.
Each Aspera telecommand can be defined as a line in the file *Mex/CmdLut/TeleCommandSLT.txt* (see appendix 6.3), the line is composed by three fields possibly followed by comments ("**;** any comment"):

ASPERA TC acronym	RPRO TC symbolic name	Parameters string

With:

ASPERA TC acronym:	Any convenient name for the telecommand, usually should be the same acronym used at ILT.
RPRO TC symbolic name:	The symbolic name of the command as reported in the CCS database.
Parameters string:	Any string of "constant" parameters

The requesting command is composed by the ASPERA TC acronym possibly followed by a string of "variable" parameters.
The program look in the *Mex/CmdLut/TeleCommandSLT.txt* file for a line beginning with the given ASPERA TC acronym. If such a line exists, the program construct and transmit a CCS telecommand request (TC-R SYM_TC) using the corresponding RPRO TC symbolic name (in field3a) and Parameters string in the variable length data field. The string of parameters in the requesting command (if any) is appended to the Parameters string.

Example:

If in *TeleCommandSLT.txt* file exists a line like:

```
ASPXXX ZS00004 R0001:=NO,R0008:=   ;just a non existent command
```

The requesting command:

```
ASPXXX   51,R0012:=6
```

is translated as a CCS telecommand request (TC-R SYM_TC) with:

TC symb. name = `ZS00004`
Data parameters = `R0001:=NO,R0008:=51,R0012:=6`

3.2.3.3 CCS Protocol Commands

These commands implement the subset of the CCS protocol [RD1] meaningful to an instrument SCOE. The command definition file is stored *in /Mex/CmdLut/Rpropkt.txt* and must not be modified by the user.
The file is an internal macro definition for CCS commands, the user just use the acronym (first field) followed (if required by the protocol) by the parameters strings. These strings are separated by spaces so never use a space inside. The acronyms may be redefined in the user macro file.

ACRONYM	**intID**	**PDUID**	**PDUsType**
CTL_CS_STATUS	4	CTL	CS_STATUS
CTL_CS_STOP	10	CTL	CS_STOP
TM-D_SPECIF_OND	1	TM-D	SPECIF_OND
TM-D_SPECIF_DFD	1	TM-D	SPECIF_DFD
TM-D_PAUSE_DFD	0	TM-D	PAUSE_DFD
TM-D_STEP_DFD	0	TM-D	STEP_DFD
TM-D_CONT_DFD	0	TM-D	CONT_DFD
TM-D_SPEED_DFD	1	TM-D	SPEED_DFD
TM-D_STOP	0	TM-D	STOP
TM_SCOE	7	TM	SCOE
COM_STATUS	5	COM	STATUS
COM_START	3	COM	START
COM_STOP	2	COM	STOP
COM_HOLD	2	COM	HOLD
COM_CONT	2	COM	CONT
TC-R_SYM_TC	3	TC-R	SYM_TC
TIME_REPORT	1	TIME	REPORT
ERR_PROTOCOL	2	ERR	PROTOCOL
ERR_UNKNOWN	2	ERR	UNKNOWN
ERR_TIMEOUT	6	ERR	TIMEOUT
USER_LOG	1	USER	LOG
USER_MMI	1	USER	MMI
SEQ	3	SEQ	XX

The RPRO CCS protocol [RD1] is formed by a number of fixed length field (ASCII and bynary), possibly followed by a variable length string preeceded by a binary field **L(b4)** indicating the number of characters in the string.
In order to communicate with CCS, the user must provide the command acronym (first column of the table above) followed by (only) the requested parameters strings separated by spaces or tabs (no spaces are allowed inside a single parameter string). Binary fields are provided as a string of numeric characters. Field **L(b4)** indicating the length of the following variable length string is computed at transmission time and must not be supplied.
As an example in order to request timely telemetry packets for APID 980 and 981 issue the following command which has only one variable length parameter string:

```
TM-D_SPECIF_OND mode=T,980,981
```

As shown on the figure on SCOE Interface, APID can be splinted as PID+CAT so that the above command can be written as:

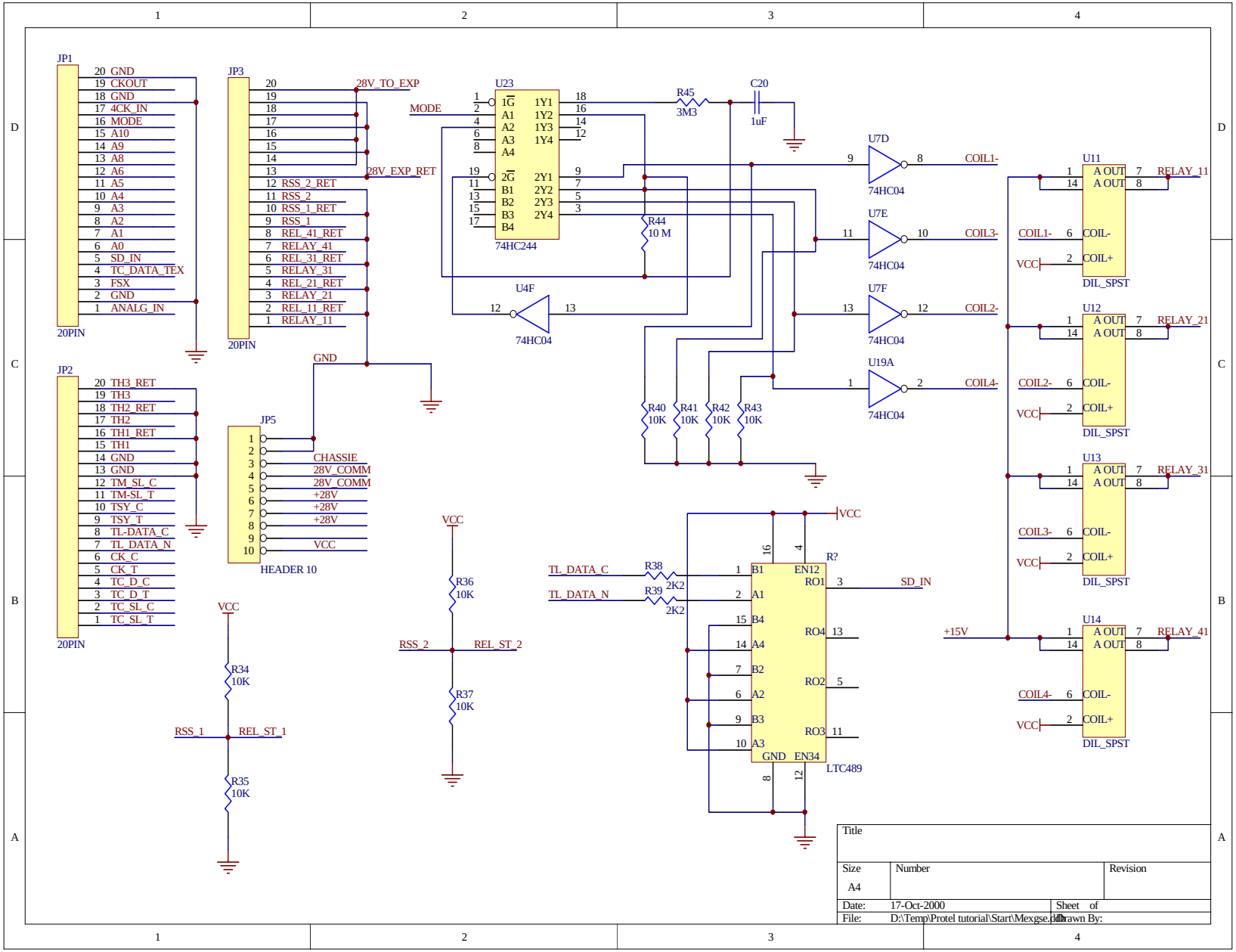
```
TM-D_SPECIF_OND mode=T,61+4,61+5
```

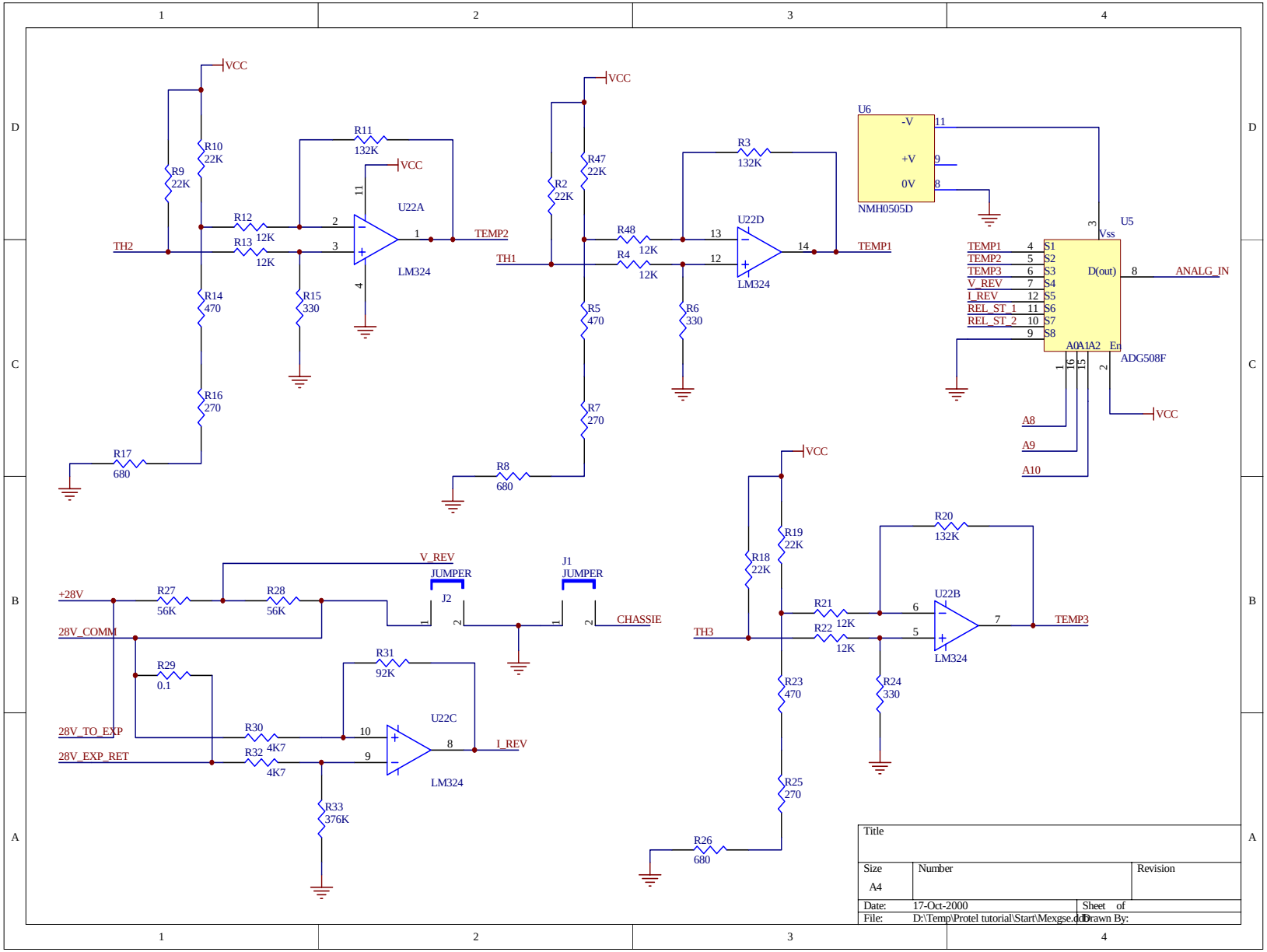
3.2.4 Commands sequence

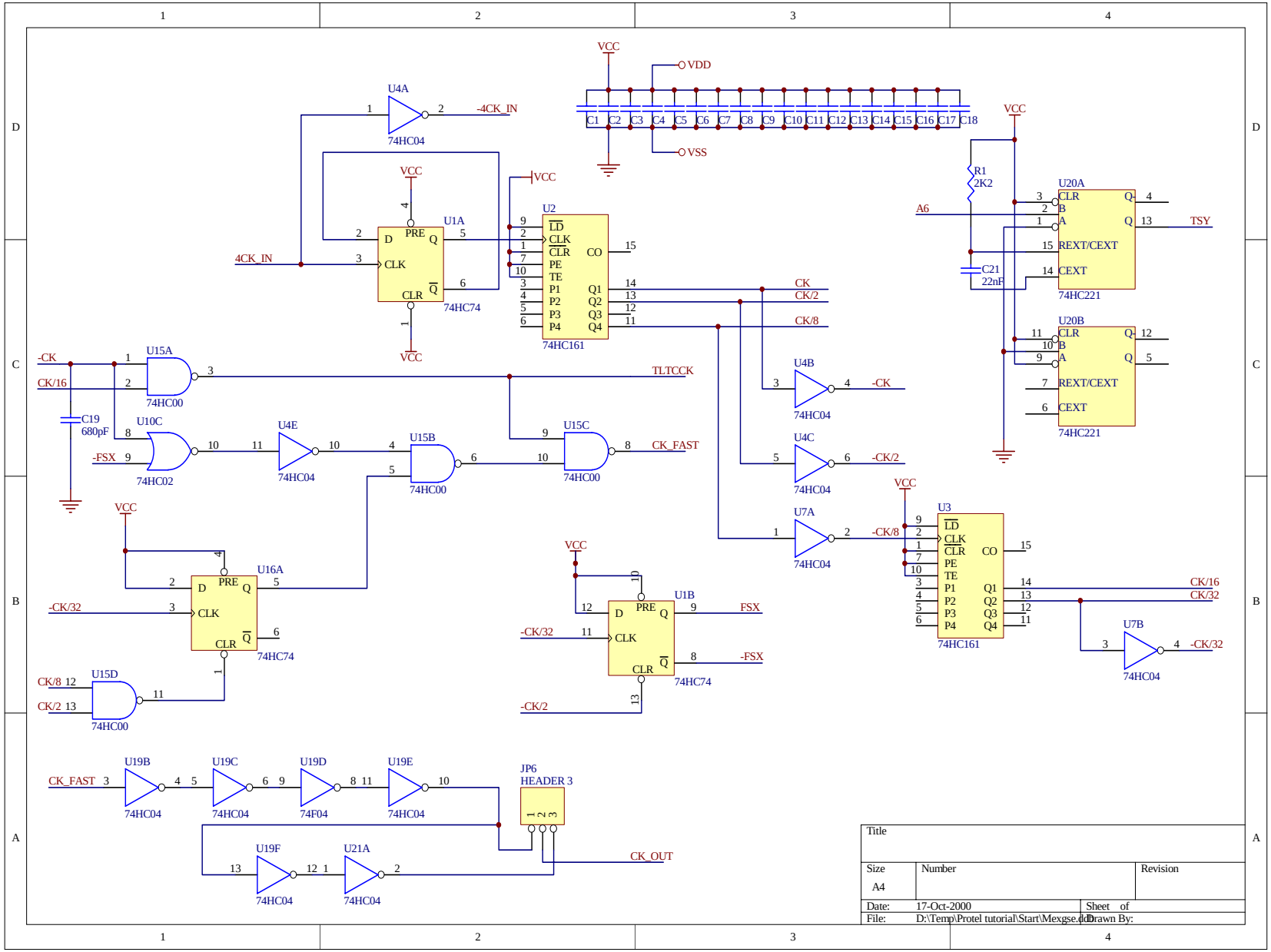
See paragraph 3.1.3.5

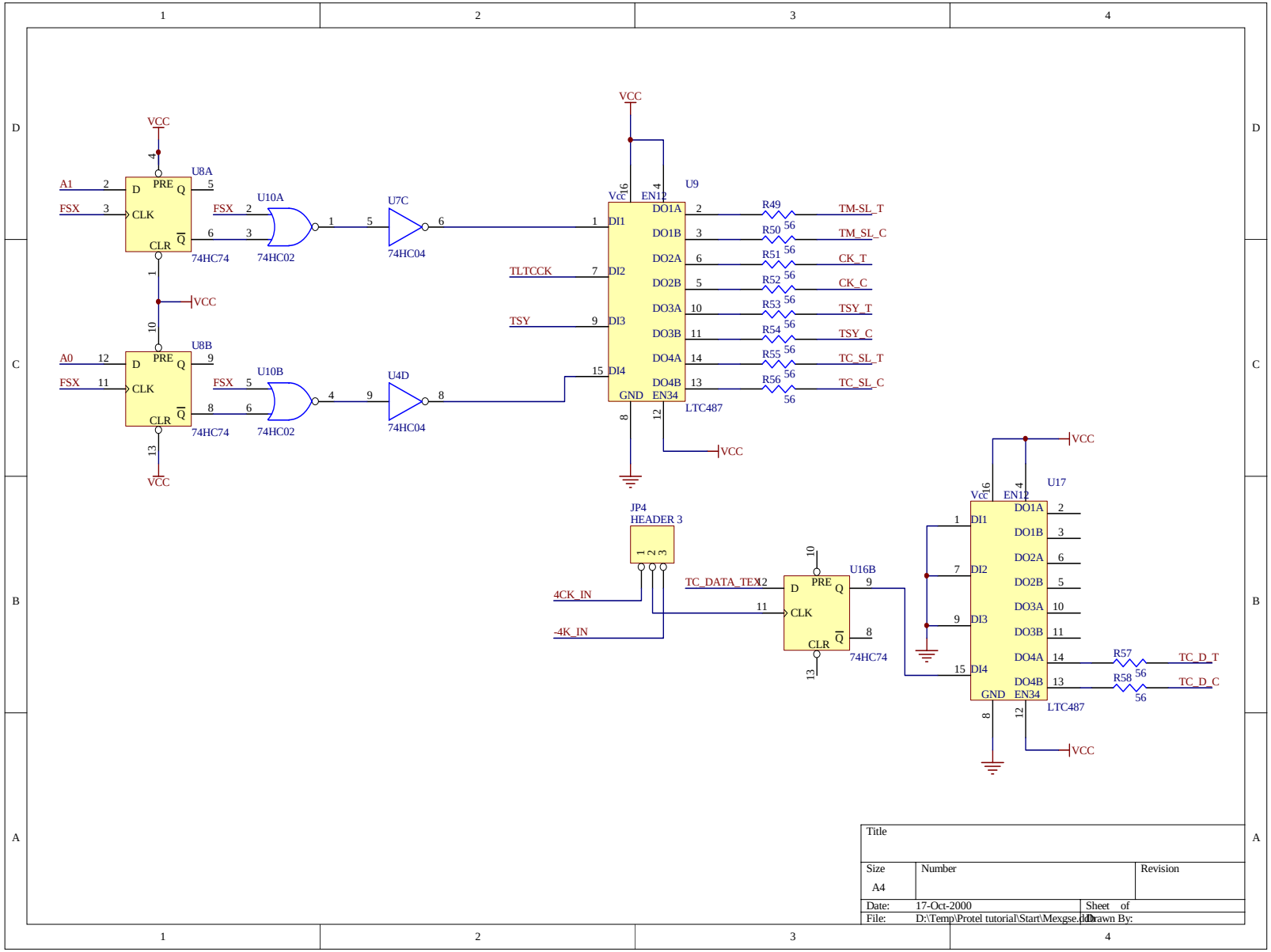
4 APPENDIX

4.1 External Electronic Schematics


JP1
20 GND
19 CKOUT
18 GND
17 4CK_IN
16 MODE
15 A10
14 A9
13 A8
12 A6
11 A5
10 A4
9 A3
8 A2
7 A1
6 A0
5 SD_IN
4 TC_DATA_TEX
3 FSX
2 GND
1 ANALG_IN
20PIN
JP3
20PIN
28V_TO_EXP
28V_EXP_RET
12 RSS_2_RET
11 RSS_2
10 RSS_1_RET
9 RSS_1
8 REL_41_RET
7 RELAY_41
6 REL_31_RET
5 RELAY_31
4 REL_21_RET
3 RELAY_21
2 REL_11_RET
1 RELAY_11
JP2
20 TH3_RET
19 TH3
18 TH2_RET
17 TH2
16 TH1_RET
15 TH1
14 GND
13 GND
12 TM_SL_C
11 TM-SL_T
10 TSY_C
9 TSY_T
8 TL-DATA_C
7 TL_DATA_N
6 CK_C
5 CK_T
4 TC_D_C
3 TC_D_T
2 TC_SL_C
1 TC_SL_T
20PIN
JP5
CHASSIE
28V_COMM
28V_COMM
+28V
+28V
+28V
VCC
HEADER 10
GND
MODE
U23
74HC244
R45
3M3
C20
1uF
R44
10 M
U4F
74HC04
U7D
74HC04
U7E
74HC04
U7F
74HC04
U19A
74HC04
COIL1-
COIL3-
COIL2-
COIL4-
R40
10K
R41
10K
R42
10K
R43
10K
U11
U12
U13
U14
A OUT
COIL-
COIL+
DIL_SPST
RELAY_11
RELAY_21
RELAY_31
RELAY_41
VCC
+15V
R34
10K
R35
10K
R36
10K
R37
10K
RSS_1
REL_ST_1
RSS_2
REL_ST_2
TL_DATA_C
TL_DATA_N
R38
2K2
R39
2K2
R?
LTC489
SD_IN
Title
Size
A4
Number
Revision
Date: 17-Oct-2000
Sheet of
File: D:\Temp\Protel tutorial\Start\Mexgse.ddb
Drawn By:


VCC
R9 22K
R10 22K
R11 132K
R12 12K
R13 12K
R14 470
R15 330
R16 270
R17 680
U22A
LM324
TH2
TEMP2
R47 22K
R2 22K
R3 132K
R48 12K
R4 12K
R5 470
R6 330
R7 270
R8 680
U22D
TH1
TEMP1
U6
-V
+V
0V
NMH0505D
U5
Vss
D(out)
ANALG_IN
TEMP1
TEMP2
TEMP3
V_REV
I_REV
REL_ST_1
REL_ST_2
S1
S2
S3
S4
S5
S6
S7
S8
A0 A1 A2 En
ADG508F
A8
A9
A10
+28V
28V_COMM
R27 56K
R28 56K
V_REV
JUMPER
J2
J1
JUMPER
CHASSIE
R29 0.1
R31 92K
28V_TO_EXP
28V_EXP_RET
R30 4K7
R32 4K7
R33 376K
U22C
LM324
I_REV
R18 22K
R19 22K
R20 132K
R21 12K
R22 12K
R23 470
R24 330
R25 270
R26 680
U22B
LM324
TH3
TEMP3
Title
Size
A4
Number
Revision
Date: 17-Oct-2000
Sheet of
File: D:\Temp\Protel tutorial\Start\Mexgse.ddb
Drawn By:


VCC
VDD
VSS
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15 C16 C17 C18
U4A
74HC04
-4CK_IN
4CK_IN
U1A
74HC74
U2
74HC161
CK
CK/2
CK/8
R1
2K2
A6
U20A
U20B
74HC221
TSY
C21
22nF
-CK
CK/16
U15A
74HC00
TLTCCK
C19
680pF
U10C
74HC02
-FSX
U4E
74HC04
U15B
U15C
CK_FAST
U4B
U4C
-CK
-CK/2
U7A
-CK/8
U3
74HC161
U16A
-CK/32
74HC74
U15D
U1B
FSX
-FSX
U7B
CK/16
CK/32
U19B
U19C
U19D
U19E
74F04
JP6
HEADER 3
U19F
U21A
CK_OUT
Title
Size
A4
Number
Revision
Date: 17-Oct-2000
Sheet of
File: D:\Temp\Protel tutorial\Start\Mexgse.ddb
Drawn By:


VCC
U8A
A1
FSX
D
PRE
Q
CLK
CLR
74HC74
U10A
74HC02
U7C
74HC04
U8B
A0
FSX
U10B
74HC02
U4D
74HC04
U9
TLTCCK
TSY
DI1
DI2
DI3
DI4
DO1A
DO1B
DO2A
DO2B
DO3A
DO3B
DO4A
DO4B
GND
EN34
LTC487
R49
R50
R51
R52
R53
R54
R55
R56
56
TM-SL_T
TM_SL_C
CK_T
CK_C
TSY_T
TSY_C
TC_SL_T
TC_SL_C
JP4
HEADER 3
4CK_IN
-4K_IN
TC_DATA_TEX
U16B
74HC74
U17
LTC487
R57
R58
TC_D_T
TC_D_C
Title
Size
A4
Number
Revision
Date:
17-Oct-2000
Sheet of
File:
D:\Temp\Protel tutorial\Start\Mexgse.ddb
Drawn By:

4.2 ASPERA High Level Command ILT

Acronym	APID	Type	Subtype	Description
APSMHKEnable	61	3	5	Enable Housekeeping report packet generation
APSMHKDisable	61	3	6	Disable Housekeeping report packet generation
ASPMTime	61	9	1	Accept time update
ASPMConnTest	61	17	1	Request connection test response
ASPMELS30	61	191	1	Turn ELS +30V on/off
ASPMELSGrid	61	191	2	Set ELS Screening Grid voltage
ASPMELSDefl	61	191	3	Set ELS Deflection Voltage
ASPMELSMcp	61	191	5	Set ELS Mcp bias
ASPMIMA30	61	191	7	Turn IMA +30V on/off
ASPMIMA12	61	191	8	Turn IMA +/- 12V on/off
ASPMIMA5	61	191	9	Turn IMA +/- 5V on/off
ASPMNPDHeat	61	191	10	Turn NPDHeaters On/Off
ASPMNPD130	61	191	11	Turn NPD1 +30V on/off
ASPMNPD1Bias	61	191	12	Set NPD1 bias voltage
ASPMNPD1Defl	61	191	13	Set NPD1 deflection voltage
ASPMNPD1Start	61	191	14	Set NPD1 Start mcp bias
ASPMNPD1Stop	61	191	15	Set NPD1 stop mcp bias
ASPMNPD230	61	191	16	Turn NPD2 +30V on/off
ASPMNPD2Bias	61	191	17	Set NPD2 bias voltage
ASPMNPD2Defl	61	191	18	Set NPD2 deflection voltage
ASPMNPD2Start	61	191	19	Set NPD2 Start surface bias
ASPMNPD2Stop	61	191	20	Set NPD2 stop surface bias
ASPMNPI30	61	191	21	Turn NPI +30 V on/off
ASPMNPIBias	61	191	22	Set NPI bias voltage
ASPMNPIDefl	61	191	23	Set NPI deflection voltage
ASPMSCAN30	61	191	25	Turn Scanner +30V on
ASPMSCANVrefmc	61	191	26	Set Scanner Vrefmc
ASPMSCANMotor	61	191	27	Turn Scanner Motor on
ASPMIMACmd	61	191	29	Send link packet to IMA
ASPMWatchDog	61	191	30	Enable watchdog
ASPMNPD1Mode	61	192	4	Set NPD1 mode
ASPMHKFreq	61	192	14	Set housekeeping packet generation frequency
ASPMPIPE	61	193	1	Send following parameters as TM
ASPMRELAX	61	193	2	Relax, ie do nothing

4.3 ASPERA High Level Command SLT

Macroname	**PKT Name**	**Parameters**	**Descritpion**
APSMHKEnable	ZAS00001	FAS00037:=	; ASP-Enable HK Report Packet Generation
APSMHKDisable	ZAS00002	FAS00037:=	; ASP-Disable HK Report Packet Generation
ASPMTime	ZAS00003	FAS00038:=	; ASP-Accept Time Update
ASPMConnTest	ZAS00004		; ASP-Connection Test Request
ASPMELS30	ZAS00100	FAS00001:=	; ASP-Turn ELS +30V ON/OFF
ASPMELSGrid	ZAS00101	FAS00002:=	; ASP-Set ELS Screening Grid Voltage
ASPMELSDefl	ZAS00102	FAS00003:=	; ASP-Set ELS Deflection Voltage
ASPMELSMcp	ZAS00103	FAS00004:=	; ASP-Set ELS MCP Bias
ASPMIMA30	ZAS00200	FAS00006:=	; ASP-Turn IMA +30V ON/OFF
ASPMIMA12	ZAS00201	FAS00007:=	; ASP-Turn IMA +/- 12V ON/OFF
ASPMIMA5	ZAS00202	FAS00008:=	; ASP-Turn IMA +/- 5V ON/OFF
ASPMNPDHeat	ZAS00300	FAS00010:=	; ASP-Turn NPD Heaters ON/OFF
ASPMNPD130	ZAS00310	FAS00011:=	; ASP-Turn NPD1 +30V ON/OFF
ASPMNPD1Bias	ZAS00311	FAS00012:=	; ASP-Set NPD1 Bias Voltage
ASPMNPD1Defl	ZAS00312	FAS00013:=	; ASP-Set NPD1 Deflection Voltage
ASPMNPD1Start	ZAS00313	FAS00014:=	; ASP-Set NPD1 Start MCP Bias
ASPMNPD1Stop	ZAS00314	FAS00015:=	; ASP-Set NPD1 Stop MCP Bias
ASPMNPD230	ZAS00320	FAS00016:=	; ASP-Turn NPD2 +30V ON/OFF
ASPMNPD2Bias	ZAS00321	FAS00017:=	; ASP-Set NPD2 Bias Voltage
ASPMNPD2Defl	ZAS00322	FAS00018:=	; ASP-Set NPD2 Deflection Voltage
ASPMNPD2Start	ZAS00323	FAS00019:=	; ASP-Set NPD2 Start Surface Bias
ASPMNPD2Stop	ZAS00324	FAS00020:=	; ASP-Set NPD2 Stop Surface Bias
ASPMNPI30	ZAS00400	FAS00021:=	; ASP-Turn NPI +30 V ON/OFF
ASPMNPIBias	ZAS00401	FAS00022:=	; ASP-Set NPI Bias Voltage
ASPMNPIDefl	ZAS00402	FAS00023:=	; ASP-Set NPI Deflection Voltage
ASPMSCAN30	ZAS00500	FAS00025:=	; ASP-Turn Scanner +30V ON
ASPMSCANVrefmc	ZAS00501	FAS00026:=	; ASP-Set Scanner V_ref_mc
ASPMSCANMotor	ZAS00502	FAS00029:=1,FAS00029:=1,FAS00027:=	; ASP-Turn Scanner Motor ON
; ASPMIMACmd	ZAS00203		; ASP-Send Link Packet to IMA
ASPMWatchDog	ZAS00005		; ASP-Enable Watchdog

ASPMNPD1Mode	ZAS00315	FAS00036,FAS00036,FAS00034:=	; ASP-Set NPD1 Mode
ASPMHKFreq	ZAS00006	FAS00031:=	; ASP-Set HK Packet Generation Frequency
; ASPMPIPE	ZAS99991		; ASP-Pipe TM: Send Parameters as TM
ASPMRELAX	ZAS99992		; ASP-Relax... ie Just Do Nothing